ASPECTS OF RADIAL BASIS FUNCTION NET DESIGN WITH APPLICATION IN SPEECH RECOGNITION

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Puranjoy Bhattacharya

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

December 1997

CERTIFICATE

Certified that the work contained in the thesis entitled "Aspects of Radial Basis Function Net Design with Application in Speech Recognition" by "Puranjoy Bhattacharya" has been carried out under our supervision and that this work has not been submitted elsewhere for a degree

(S K Mullick)
Professor Emeritus,
Department of Electrical Engineering,
I.I.T Kanpur

(D Kundu)
Associate Professor,
Department of Mathematics,
I.I.T Kanpur

(V.P Sinha)
Professor,
Department of Electrical Engineering,
I I T Kanpur.

Synopsis

This thesis addresses the problem of *pattern recognition* (PR) using a class of neural networks called the Radial Basis Function (RBF) Net. The problem of PR arises in an effort to imbue cognitive capabilities into digital computers. Most of us realise, for example, how advantageous it would be if a word-processing package would allow verbal entry of data. This would require segments of the transduced speech signal to be classified into the corresponding acoustic events; where each acoustic event could signify a phoneme, which is the simplest acoustically distinguishable portion of an utterance, a syllable, a word, or even a sentence. It is this process of classification that forms the essence of PR.

PR mechanisms, like the one required for the speech-to-text problem are, however, extremely difficult to automate. The primary reason for this is that we still have very little idea about how the human brain actually stores and manipulates data. The engineering approach to PR, therefore, heavily relies on existing data structures and models — which happen to be diverse — in realising pattern classifiers. The processing models expanded upon in this thesis possess the following properties :

- For a fixed given set of items (each item comprising of measurements made on the entities to be classified), these models are capable of reproducing the class information with arbitrary precision in case no bounds are placed on their size. This makes them suitable for application in a fairly large class of PR applications.

- These models allow their classification performance to be traded against their size. An improvement in the error performance (meaning fewer misclassifications) can usually be achieved at the cost of increased computational and

storage costs.

- Within limits, these models allow their internal data representations to be interpreted, so that these can be modified or constrained using apriori information available to the user about the problem from other sources.

This synopsis is aimed at providing a glimpse of some of the crucial ideas in the thesis, apart from summarising its contents. The description of the notations used is followed by a brief introduction to the class of Artificial Neural Network models, of which the RBF net is an instance. After this, a statement of the design problem is made, along with an exposition of the structure of the RBF net. It is then shown as to how the RBF model fits into a PR formulation with an appropriate specification of the input and output spaces. Then the topic moves on to template-matching, which is one of the central concepts in the thesis. This is finally followed by a mention of the focal points of the thesis and a chapterwise summary of the same.

Notation

In this thesis, the most common notations used are as in Table 1.

Neural PR

Artificial Neural Networks (ANN) provide an emerging paradigm in pattern recognition. The field of ANN encompasses a large variety of models, all of which have two important characteristics :

1. They are composed of a large number of structurally and functionally similar units called neurons usually connected in various configurations by weighted links.

2. The ANN model parameters are derived from supplied input-output paired data sets by an estimation process called training.

Description	Example(s)
Normal, lower case italics for *scalars*	x
Bold, lower case italics for *vectors*	$\boldsymbol{x}$
Bold, upper case italics for *matrices*	$\boldsymbol{X}$
Upper case italics for *transformations*	T
Calligraphic upper case for *sets*	$\mathcal{C}$
Normal phi or psi for scalar functions of one variable	ϕ, ψ
Bold phi or psi for vectors whose elements are scalar functions of one variable	$\boldsymbol{\phi}$, $\boldsymbol{\psi}$
Normal eta or gamma for constants	η, γ
Transpose of a vector $\boldsymbol{x}$	$\boldsymbol{x}^T$
Transpose of a matrix $\boldsymbol{X}$	$\boldsymbol{X}^T$
Inverse of a square matrix $\boldsymbol{X}$	$\boldsymbol{X}^{-1}$
Pseudo-inverse of a matrix $\boldsymbol{X}$	$\boldsymbol{X}^+$
Norm of a vector $\boldsymbol{x}$	$\|\|\boldsymbol{x}\|\|$
L_2 norm of a vector $\boldsymbol{x}$	$\|\|\boldsymbol{x}\|\|_2$
Froebenius norm of a matrix $\boldsymbol{X}$	$\|\|\boldsymbol{X}\|\|_F$
Inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$	$< \boldsymbol{x}, \boldsymbol{y} >$
Estimate of $\boldsymbol{x}$	$\hat{\boldsymbol{x}}$

Table 1: List of Notations

ANNs had been developed as a class of trainable model-free estimators in an attempt to emulate the working of naturally intelligent systems (e.g. the human brain). The RBF model, which is the focus of this thesis, is an ANN. For a given kind of neurons forming the ANN, the configuration and the training paradigm are what crucially determine how well the ANN performs for a specific PR problem, and these issues will be considered central to this thesis.

Aim

In this thesis, the model that is used for PR is an ANN known as the *Radial Basis Function (RBF) Net*, which represents a nonlinear map

$$\boldsymbol{y} = \sum_{i=1}^{p} \boldsymbol{w}_i \phi_i(||\boldsymbol{x} - \boldsymbol{\mu}_i||_2)$$

where, $y \in \mathsf{R}^m$ is a vector whose elements reflect the output class membership values (i.e. the i^{th} element being equal to 1 or 0 corresponds to the confirmation or nullification respectively of the hypothesis that the i^{th} class is present) and $\boldsymbol{x} \in \mathsf{R}^n$ is a vector whose elements are input measurements to be used for classification. $\boldsymbol{w}_i$, the weight vectors and $\boldsymbol{\mu}_i$, the prototype vectors used in the model have to be derived through training. ϕ_i represent (nonlinear) continuous monotonically non-increasing nonnegative functions of one variable defined over the range $[0, \infty)$ which have a finite second moment

$$\sigma_i = \frac{\int_0^\infty q^2 \phi_i(q) dq}{\int_0^\infty \phi_i(q) dq} .$$

To satisfy the last criterion, it is necessary for these functions to decay rapidly to zero with increasing value of the independent variable.

RBF net parameters like $\boldsymbol{\mu}_i$ and $\boldsymbol{w}_i$ are usually obtained through *training*, whereby these vectors are chosen so that the classification error for the model is minimised over a set of input measurement-output class paired data supplied apriori, known as the *training set*. By the design of an RBF net is meant a process of placing constraints on the structure and the training procedure of RBF nets so that the following objectives are satisfied :

- The classification error (or misclassification rate) is minimised.
- The number of model parameters are minimised to economise storage and computational costs.
- The model is modular in the sense that it can be decomposed into substructures which are structurally distinguishable, and whose functional domains are roughly known.

In this thesis, there are two main objectives :

1. To develop techniques for structuring and training RBF nets so as to satisfy the design criteria stated above.
2. To seek an appropriate method for the solution of the problem of artificial speech recognition (ASR) within the RBF framework.

The Generalised PR Scheme and the RBF

The problem of classification of data can be formulated as one of making a decision, based on premises derived from a series of measurements. Mathematically, the question is one of deriving a map

$$T : \mathcal{M} \mapsto \mathcal{C}$$

where $\mathcal{M}$ is the universe of measurements which might include categorical or numerical variables, and $\mathcal{C}$ is the set of classes.

It is possible to quantise the measurements, and reformulate the complete pattern recognition problem in the categorical (or symbolic) context. A parallel approach involves converting the inputs as well as the outputs into numerical values, wherever necessary, and formulating the classification problem in a metric space of reals. Categorical variables can readily be mapped onto the vertices of a unit hypercube in an Eucleidian space called the *class space*, without inconsistencies in accompanying interpretations. To illustrate how this is done, let us assume that there are 4 classes at the output of the classifier; these can be encoded as follows :

CLASS	CODE
Class 1	[1 0 0 0]
Class 2	[0 1 0 0]
Class 3	[0 0 1 0]
Class 4	[0 0 0 1]

As apparent, the encoding vectors are mutually orthogonal (in a 4-D Eucleidian space). The problem of PR, thus, can be formulated as one of realising a nonlinear vector function

$$T : \mathrm{R}^n \mapsto \mathrm{R}^m$$

The notion of dissimilarity in this setting can be realised through the notion of metrics or pseudometrics suitably chosen for the input and output spaces. In this thesis, an RBF net is used to realise the function T; the fact that the RBF net belongs to the class of *universal approximators*, i.e. it can approximate any given

function to an arbitrary degree of accuracy if allowed a sufficient number of internal nodes, provides one of the rationalisations for this choice.

Template Matching : An Important Concept

One of the central ideas of pattern recognition is that of *template matching.* Quoting K.S. Fu, one of the pioneers in the field of PR,

> "An intuitively appealing approach for pattern recognition is the approach of 'template-matching'. In this case, a set of templates or prototypes, one for each pattern class, is stored in the machine. The input pattern (with unknown classification) is compared with the template of each class, and the classification is based on a preselected matching criterion or similarity criterion. In other words, if the input pattern matches the template of the i^{th} pattern class better than it matches any other templates, then the input is classified as from the i^{th} pattern class."

Template matching surfaces in various garbs for different data representations in practical PR implementations. In this thesis, it is shown how an RBF net can be interpreted as a two-level template matching operation, so that a substantial part of the design of an RBF net can be reduced to forming templates of different kinds. The first level of template-matching is accomplished by the Gaussian functions in the RBF net :

$$\begin{aligned} \phi(d_i) &= e^{-kd_i^2} \\ where, \quad d_i &= distance(sample, prototype_i) \ . \end{aligned}$$

It is evident that the Gaussian function implements a distance-based similarity measure. In the case where the inputs belong to a measurement space in R^n, the templates (or prototypes) are frequently vectors in R^n. With regard to speech recognition, which is a subproblem tackled in this thesis, the basic design aspect developed is how to make speech templates for this level of the RBF that are invariant to time and frequency variations.

The second level of templates occur embedded in the subsequent linear map in the RBF. Each element of the output class vector $\boldsymbol{y}$ is evolved as

$$y_i = < \boldsymbol{w}_i, \boldsymbol{z} >, \quad i = 1, \ldots, m$$

where $\boldsymbol{z}$ is the intermediate vector formed by stacking the outputs of the Gaussian basis functions. The inner product implements a correlation-based similarity measure between $\boldsymbol{z}$ and the prototype weight vector $\boldsymbol{w}_i$. As part of this thesis, this interpretation is used significantly in modularising the linear part of the RBF.

Focal Points of the Thesis

The thesis has three focal points as follows :

Selection of Basis Functions : Under this, various techniques for selecting basis functions for use in the RBF net, and their impact on the error-complexity trade-off is studied.

Modularisation of RBF Nets : This topic deals with methods to convert RBF nets into modular ANNs. Modularity is enforced both at the level of the basis functions and the subsequent linear map, leading to a set of possible configurations.

Making Templates for Speech Recognition : Since the speech signal is prone to translations and dilations in time and frequency, a stepwise procedure is outlined so that templates for use in the RBF network are obtained which are invariant to the above-mentioned variations.

Layout of the Thesis

The thesis is organised into eight chapters, the first of which is the introduction.

The second chapter is mainly a review of the literature on RBF nets. The main fact that is sought to be established in this chapter is that the RBF model can be visualised to be the confluence of ideas in several inter-related fields, and that

it can be accordingly interpreted in a variety of ways. The development of RBF (or RBF analogue) models from the angles of approximation, regression and fuzzy decision-making are described in detail. The basic techniques for training RBFs using gradient-descent and cluster-and-pseudoinverse are also discussed; it is argued that the former amounts to iterative minimisation of a global error criterion, while the latter involves model optimisation based on an interpretation of the internal structure of the RBF, and therefore, must be preferred. Choice of the latter training model also allows meaningful design of the RBF net in terms of the objectives mentioned earlier.

The third chapter dwells on the effects of two important preprocessing operations on the output of the basis function layer viz., thresholding (with a small threshold value) and normalisation, on the performance of the RBF net. It is experimentally shown that both improve error performance. With regard to the former, an explanation is provided from the point of view of enforcing a compact support for basis functions, and from the angle of the bias-variance trade-off. The performance enhancement with normalisation is sought to be explained first as the result of a soft-maximisation operation, and next, as a spin-off of the correlation-based linear template matching operation described earlier. The background material corresponding to the two-level match interpretation and the softening of frequently used nonlinear functions is also provided in this chapter.

Chapter 4 deals with the error-complexity trade-off. There are two dominant ideas — first that error and complexity minimisation usually work against each other, and second that the problem of basis function selection is equivalent to the problem of variable selection in linear regression. The second idea implies the direct application of available methods in statistics for the purpose of basis selection. Corresponding to the case of no penalty on model complexity, the orthogonal least squares technique available in literature is cited. Incorporating the complexity minimising requirement into the design can be performed in two ways. The first approach involves converting the complexity into an error penalty, as in the Akaike Information Criterion, and the second, converting the error into a complexity penalty. For the latter, a new information criterion is proposed which penalises

equipartitioning of the regressional sum of squares by basis functions, and substitutes the error sum of squares by an equipartitioned (partial) regressional sum of squares. The method is also shown to have interesting extensions in the case where the basis functions can be organised into a hierarchy, such that the parent basis functions can be obtained as a linear combination of the children basis functions.

The next two chapters are on techniques for modularising the RBF net. In the first, the basis functions are used as the vehicle of modularisation. This implies that a structure has to be imposed on the basis functions during the clustering stage. Two such structures are used — the tree structure and the flat grid structure. The former uses the LBG vector-quantisation algorithm developed by Linde, Buzo and Gray, while the latter uses Kohonen's Self-Organising-Map (SOM) for clustering with the respective structural constraints. In the former case, entry into nodes at a certain level in the tree, and in the latter, the nearest neighbour quantiser node are used as the gating functions for the respective modules.

In Chapter 6, modularity is imposed on the RBF net by replacing the linear map by piecewise linear maps. Once again, there are two structures imposed — the tree and the flat grid. The former develops into a regression tree (the 1-dimensional analogue of which is found in literature) while the latter takes the shape of a set of unit-rank matrices which are trained by an SOM-like algorithm; the inner product (of the intermediate vector z) with the rightmost vector v in the unit rank decomposition

$$W = uv^T$$

serves as the gating function in either case. In both chapters, the hierarchical structures are found to produce slightly inferior performance, which is sought to be explained as a result of the accompanying hard binary splits at each level.

Chapter 7 deals with the question of how the speech recognition problem can be accommodated within the RBF framework. For this, the template interpretation is brought to the forefront, and it is explained that the crux of the problem is in obtaining speech templates that are invariant with respect to translation and dilation in time and frequency. A step-by-step procedure for the conversion of speech signal segments into a sequence of spectral vectors is logically developed,

and it is illustrated as to how, first by adopting a perceptual mel scale, and later by computing the cepstral coefficients, the effects of the aforesaid transformations in frequency are nullified.

The rest of the chapter is devoted to the problem of deriving a representation for a sequence of vectors that is relatively invariant to translations and dilations in the time index. The important concept developed in this regard is that of forming distance-based tree representations for waveforms. The idea stems from the effort to identify acoustic events as sets of closely-valued points contiguous in time. The implementation of the tree formation procedure is akin to single-linkage clustering with a constraint on the time index. Once the tree is formed, the sequence of nodes at a particular depth is used to represent the original waveform. The representation derives templates which are vector sequences of a fixed length, and have been found to be relatively invariant to monotonic transformations of the time index. These are subsequently used in speech recognition for distinguishing between six closely related diphones. Although the recognition performance for a one-shot developmental scheme was not comparable with that of existing popular implementations, the ease, elegance, and minimal computational and storage requirements of the model certainly warrant attention for this new model.

The last chapter contains a summary of important conclusions and the scope of future work on some of the aspects of the research effort described in the thesis.

Acknowledgements

I wish to express my deep gratitude to my supervisors. They have constantly stimulated me with creative material, encouraged me to think freely, provided moral and material support throughout my work and allowed me a closeness which I greatly cherish. Dr.Mullick's help has transcended the academic; having got married toward the end of the programme, I even subsist on his furniture. During periods of stress, I have freely communicated with my supervisors, and each time, they have stood by me As teachers, they have been excellent. Dr. Kundu has been responsible for reviving my flagging interest in statistics. My thanks to Dr. Sinha for readily agreeing to take up the role of being my administrative supervisor after Dr. Mullick's retirement. I owe my interest in symbolic processing and in the general structure of object representations to a course that he taught during my Master's curriculum.

I must also take this occasion to thank all my other teachers in the Department of Electrical Engineering and the Department of Mathematics, IIT Kanpur, in whose tutelage I've been nurtured since my bachelor days. They have maintained a culture of openness and accessibility that has allowed the students to grow. They have extended their help unconditionally whenever required. Working as a teaching assistant under Dr.Sharan for electronics labs has been a pleasure. On one hand, it was a revelation on the art of electronic design; on the other, he was never hesitant to let me off when I needed it badly. I owe a great deal to Dr. Sumana Gupta; I have constantly been using lab facilities that are funded by her projects. I also wish to thank Dr. G.Sharma and Dr. A.Mahanta for allowing me to use the DSP lab, and Dr. S.Umesh for providing the vital TIMIT Database CDROM.

Outside IIT Kanpur, I wish to thank Dr. S.S.Aggrawal, CEERI Delhi for providing me with recorded Hindi speech data and Dr. Scott Fahlman and Dr. Arun

Kumar, IIT Delhi, for emailing me the Deterding Vowel Data Set at a point of uncertainty about the continuation of Internet facilities in IIT Kanpur.

My wife has been a great help throughout the last phase of my work. She has taken time off from own busy schedules to run the house entirely on her own. My friends have helped me whenever I needed them. I particularly wish to thank Manish Shrikhande for proof-reading my thesis. Thanks are also due to Gomathi Shankar for accommodating me on the various lab machines.

Finally, I wish to thank all my friends, especially Sudipto Mukhopadhyay, Joydeb Mukherjee, Satyavrat Patnaik, R.Jitendra Das, Milind Patil, Arun Kumar, Indra Narayan Kar, Kushal Kanungo, Apu Sivadas, T.Sajith, Arun Saha, Alok Srivastav, Shreos Roychowdhury, Shushanto Mahapatra, T.K.Roy, Shantanu Banerjee, Suvrat Gupta, Bhaskar Bhar, Sobha Nair and Pratima Aggrawal for making life so much livable in IIT Kanpur.

Contents

Chapter 1

Introduction

This thesis addresses the problem of *pattern recognition* (PR) using a class of neural networks called the Radial Basis Function (RBF) Net. The problem of PR arises in an effort to imbue cognitive capabilities into digital computers. Most of us realise, for example, how advantageous it would be if a word-processing package would allow verbal entry of data. This would require segments of the transduced speech signal to be classified into the corresponding acoustic events; where each acoustic event could signify a phoneme, which is the simplest acoustically distinguishable portion of an utterance, a syllable, a word, or even a sentence. It is this process of classification that forms the essence of PR. Quoting K.S. Fu [1], one of the pioneers in the field of PR [1],

> "The problem of pattern recognition usually denotes a discrimination or classification of a set of processes or events."

PR mechanisms, like the one required for the speech-to-text problem are, however, extremely difficult to automate. The primary reason for this is that we still have very little idea about how the human brain actually stores and manipulates data. The engineering approach to PR, therefore, heavily relies on existing data structures and models — which happen to be diverse — in realising pattern classifiers. The processing models expanded upon in this thesis will be shown to possess the following properties :

- For a fixed given set of items (each item comprising of measurements made on the entities to be classified), these models are capable of reproducing the class information with arbitrary precision in case no bounds are placed on their size. This makes them suitable for application in a fairly large class of PR applications.

- These models allow their classification performance to be traded against their size. An improvement in the error performance (meaning fewer misclassifications) can usually be achieved at the cost of increased computational and storage costs.

- Within limits, these models allow their internal data representations to be interpreted, so that these can be modified or constrained using apriori information available to the user about the problem from other sources.

This chapter is aimed at providing an overview of the thesis, along with a glimpse of some of the crucial ideas. The following section provides a description of the notations used in the thesis. This is followed by a brief introduction to the basic PR paradigms, with particular stress on the class of Artificial Neural Network (ANN) models, of which the RBF net is an instance. After this, a statement of the design problem is made, along with an exposition of the structure of the RBF net. It is then shown as to how the RBF model fits into a PR formulation with an appropriate specification of the input and output spaces. The next section illustrates how the RBF net can be applied to adaptive signal processing, thereby underlining the versatility of the model. Then the topic moves on to template-matching, which is one of the central concepts in the thesis. This is finally followed by a mention of the focal points of the thesis and a chapterwise summary of the same. The last section in the chapter provides the rationale for the choice of the data sets used in this thesis.

1.1 Notation

In this thesis, the most common notations used are as in Table 1.1.

Description	Example(s)
Normal, lower case italics for *scalars*	x
Bold, lower case italics for *vectors*	$\boldsymbol{x}$
Bold, upper case italics for *matrices*	$\boldsymbol{X}$
Upper case italics for *transformations*	T
Calligraphic upper case for *sets*	$\mathcal{C}$
Normal phi or psi for scalar functions of one variable	ϕ, ψ
Bold phi or psi for vectors whose elements are scalar functions of one variable	$\boldsymbol{\phi}$, $\boldsymbol{\psi}$
Normal eta or gamma for constants	η, γ
Transpose of a vector $\boldsymbol{x}$	$\boldsymbol{x}^T$
Transpose of a matrix $\boldsymbol{X}$	$\boldsymbol{X}^T$
Inverse of a square matrix $\boldsymbol{X}$	$\boldsymbol{X}^{-1}$
Pseudo-inverse of a matrix $\boldsymbol{X}$	$\boldsymbol{X}^{+}$
Norm of a vector $\boldsymbol{x}$	$\|\|\boldsymbol{x}\|\|$
L_2 norm of a vector $\boldsymbol{x}$	$\|\|\boldsymbol{x}\|\|_2$
Froebenius norm of a matrix $\boldsymbol{X}$	$\|\|\boldsymbol{X}\|\|_F$
Inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$	$< \boldsymbol{x}, \boldsymbol{y} >$
The span [0 1] on the real line	I
Estimate of $\boldsymbol{x}$	$\hat{\boldsymbol{x}}$
Residual sum of squares with p variables	$\mathcal{E}_p$

Table 1.1: List of notations commonly used in the thesis.

1.2 Basic Approaches to Pattern Recognition : A Brief Mention

PR applications have been varied, and so also the associated data structures and processing paradigms. In the course of time, four significant approaches to PR have evolved These are [2] :

Statistical PR : Here, the problem is posed as one of composite hypothesis testing, each hypothesis pertaining to the premise of the datum having originated from a particular class; or as one of regression from the space of measurements to the space of classes. The statistical method for solving the same involves the computation of the class conditional probability densities, which remains the main hurdle in this approach. It will be shown in section 2.2 that this subproblem has an important link to the RBF model discussed in this thesis. The statistical approach to PR is one of the oldest, and is still widely used [3].

Syntactic PR : In syntactic pattern recognition, each pattern is assumed to be composed of subpatterns or *primitives* strung together in accordance with the generation rules of a grammar characteristic of the associated class. Class identification is accomplished by way of parsing operations using automata corresponding to the various grammars [4, 5] Parser design and grammatical inference are two difficult issues associated with this approach to PR and are responsible for its somewhat limited applicability.

Knowledge-Based PR : This approach to PR [6] is evolved from advances in rule-based systems in artificial intelligence (AI). Each rule is in form of a clause that reflects evidence about the presence of a particular class. The main subproblems spawned by the methodology are

1. how the rule base may be constructed, and
2. what mechanism might be used to integrate the evidence yielded by the invoked rules.

In this thesis, section 2.3 shows how an RBF can be regarded as an implementation of a knowledge-based system using fuzzy logic for evidence integration.

Neural PR : Artificial Neural Networks (ANN) provide an emerging paradigm in pattern recognition. The field of ANN encompasses a large variety of models [13], all of which have two important characteristics :

1 They are composed of a large number of structurally and functionally similar units called neurons usually connected in various configurations by weighted links.

2 The ANN model parameters are derived from supplied input-output paired data sets by an estimation process called training.

ANNs had been developed as a class of trainable model-free estimators in an attempt to emulate the working of naturally intelligent systems (e.g. the human brain). The RBF model, which is the focus of this thesis, is an ANN. For a given kind of neurons forming the ANN, the configuration and the training paradigm are what crucially determine how well the ANN performs for a specific PR problem, and these issues will be considered as crucial in this thesis.

1.3 Aim

In this thesis, the model that is used for PR is an ANN known as the *Radial Basis Function (RBF) Net*, which represents a nonlinear map

$$\boldsymbol{y} = \sum_{i=1}^{p} \boldsymbol{w}_i \phi(||\boldsymbol{x} - \boldsymbol{\mu}_i||_2) \tag{1.1}$$

where $y \in \mathsf{R}^m$ is a vector whose elements reflect the output class membership values (i.e. the i^{th} element being equal to 1 or 0 corresponds to the confirmation or nullification respectively of the hypothesis that the i^{th} class is present) and $\boldsymbol{x} \in \mathsf{R}^n$ is a vector whose elements are input measurements to be used for classification. $\boldsymbol{w}_i$, the weight vectors and $\boldsymbol{\mu}_i$, the prototype vectors used in the model have to

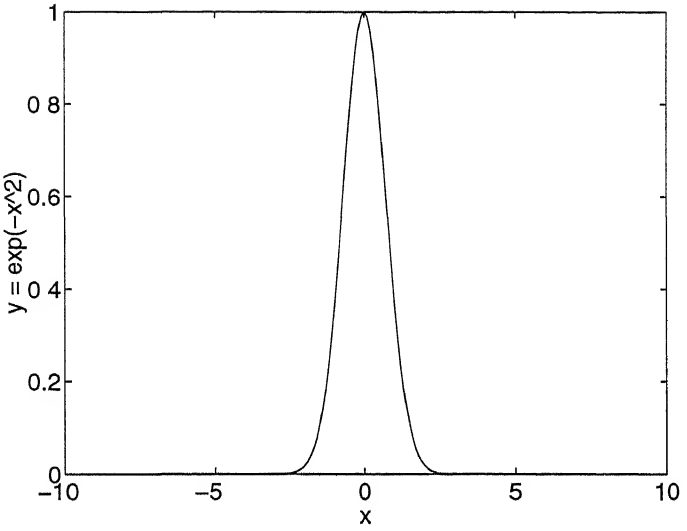

Figure 1 1: The Gaussian function.

be derived through training. ϕ represents a (nonlinear) continuous monotonically non-increasing nonnegative function of one variable defined over the range $[0, \infty)$ which has a finite second moment (like the right half of the Gaussian in Figure 1.1) given by

$$\sigma_\phi = \frac{\int_0^\infty q^2 \phi(q) dq}{\int_0^\infty \phi(q) dq} \ . \tag{1.2}$$

To satisfy the last criterion, it is necessary for these functions to decay rapidly to zero with increasing value of the independent variable[1].

Before proceeding further, it should be mentioned that the equation 1.1 can be rewritten as :

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{W}\boldsymbol{z} \qquad (1.3) \\ where \quad \boldsymbol{z} &= \begin{bmatrix} z_1 \\ \vdots \\ z_p \end{bmatrix} \\ z_i &= \phi(||\boldsymbol{x} - \boldsymbol{\mu_i}||_2) \\ and \quad \boldsymbol{W} &= [\boldsymbol{w}_1 \cdots \boldsymbol{w}_p] \ . \end{aligned}$$

[1] An explicit criterion for locality of the basis functions will be introduced later in the thesis. The current statement is, however sufficient for all subsequent interpretations.

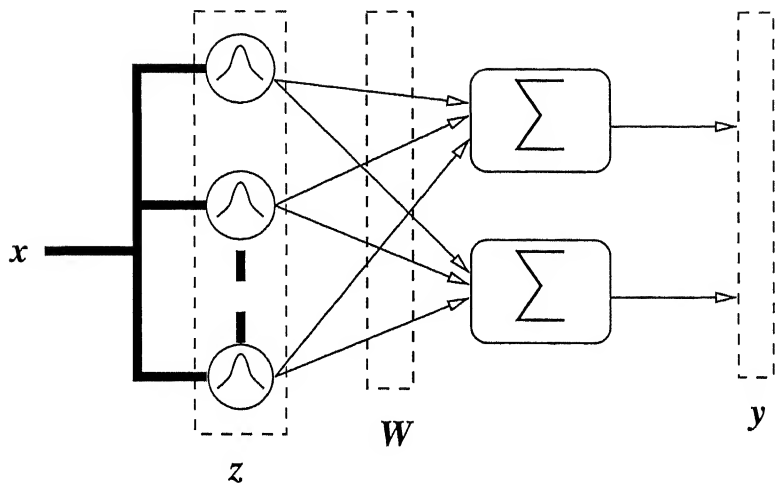

Figure 1.2· Pictorial depiction of the RBF net

This form of the RBF is pictorially depicted in Figure 1.2.

RBF net parameters like μ_i and w_i are usually obtained through *training*, whereby these vectors are chosen so that the classification error for the model is minimised over a set of input measurement-output class paired data supplied apriori, known as the *training set*. By the design of an RBF net is meant a process of placing constraints on the structure and the training procedure of RBF nets so that the following objectives are satisfied :

- The classification error (or misclassification rate) is minimised.
- The number of model parameters are minimised to economise storage and computational costs.
- The model is modular in the sense that it can be decomposed into substructures which are structurally distinguishable, and whose functional domains are roughly known.

In this thesis, there are two main objectives :

1. To develop techniques for structuring and training RBF nets so as to satisfy the design criteria stated above.
2. To seek an appropriate method for the solution of the problem of artificial speech recognition (ASR) within the RBF framework.

1.4 The Generalised PR Scheme and the RBF

The problem of classification of data can be formulated as one of making a decision, based on premises derived from a series of measurements. Mathematically, the question is one of deriving a map

$$T \cdot \mathcal{M} \mapsto \mathcal{C}$$

where $\mathcal{M}$ is the universe of measurements which might include categorical or numerical variables, and $\mathcal{C}$ is the set of classes. However, the number of ways in which such a map can be derived are potentially infinite, and further constraints need to be imposed on the realisation of T to prevent the problem from becoming ill-posed Frequently, the constraints are in terms of restricting the map T to a single realisable family of functions $\mathcal{T}$ which are consistent with respect to the physical reality contextual to the problem at hand. The problem of classifier design then reduces to model optimisation in terms of reducing the mismatch to a supplied data set comprising measurements and associated class labels.

It is important to realise at this juncture that the measurements input to the PR system may be categorical (i.e. symbolic) or numerical (i.e. belong to the field of reals R or complex numbers C). In case all the inputs are categorical, Boolean or syntactic models are used by the PR system. For numerical or mixed inputs, however, using the above models will require a prior quantising mechanism. The other approach involves converting the inputs as well as the outputs into numerical values, wherever necessary, and formulating the classification problem in a metric space of reals. Categorical variables can readily be mapped onto the vertices of a unit hypercube in an Euclidean space called the *class space*, without inconsistencies in accompanying interpretations. To illustrate how this is done, let us assume that there are 4 classes at the output of the classifier; these can be encoded as shown in Table 1.2. As apparent, the encoding vectors are mutually orthogonal (in a 4-D Euclidean space).

The problem of PR, thus, can be formulated as one of realising a nonlinear vector function

$$T : \mathsf{R}^n \mapsto \mathsf{R}^m \ . \tag{1.4}$$

CLASS	CODE
Class 1	[1 0 0 0]
Class 2	[0 1 0 0]
Class 3	[0 0 1 0]
Class 4	[0 0 0 1]

Table 1.2: Encoding classes with unit vectors.

The notion of dissimilarity in this setting can be realised through the notion of metrics or pseudometrics suitably chosen for the input and output spaces.

The idea of template matching often spawns, in a PR system, an intermediate space called the *pattern space* [2]. This is based on the assumption that it is possible to obtain a finite set of representative vectors belonging to the input space of measurements called templates or prototypes, which are distinct, but may share close values with respect to certain attributes (elements). The degrees of similarity of an input in the measurement space to each of the templates may then be quantified, and the resulting values bunched together to form a *pattern vector*; the vector space which is the linear span of all pattern vectors ($\subset \mathsf{R}^p$) obtained in this manner will be called the pattern space $\mathcal{A}$. The pattern recognition operation under this setup is thus split into two processes :

1. Computing the degree of closeness of an input pattern to each member of the template set to comprise the pattern vector.

2. Constructing a map from the pattern space to the output class space.

Mathematically, therefore, the original map T has been split into

$$F : \mathcal{M} \mapsto \mathcal{A}$$

and

$$L : \mathcal{A} \mapsto \mathcal{C}.$$

Under the assumption that F can be realised through a set of Gaussian functions centred at the templates, and L is a linear map from $\mathsf{R}^p \mapsto \mathsf{R}^m$, the above formulation

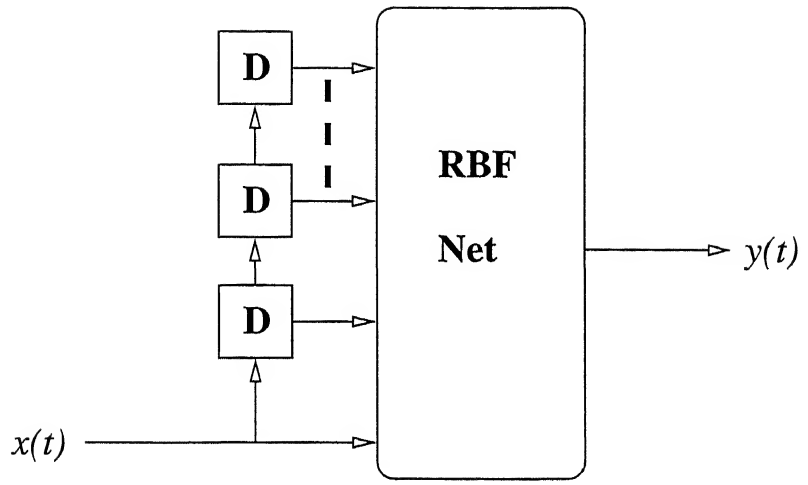

Figure 1.3: Implementation of a time-invariant nonlinear filter using the RBF.

takes the shape of a RBF net. This sums up the way RBFs are directly linked to PR via the template-matching paradigm.

In this thesis, an RBF net is used to realise the function T; the fact that the RBF net belongs to the class of *universal approximators*, i.e. it can approximate any given function to an arbitrary degree of accuracy if allowed a sufficient number of internal nodes, provides yet another rationalisation for this choice.

1.5 RBF as a Trainable Nonlinear System

The RBF net has an attractive structure from the point of view of nonlinear system implementation. On one hand, an RBF net is a universal approximator; on the other hand, if one should discount the layer of basis functions, the RBF would become a linear function. It is easy to convert an RBF into a nonlinear time-invariant system by feeding the input from a tapped-delay line as shown in Figure 1.3.

Furthermore, since all of the elements of the intermediate vector input to the linear transformation in an RBF net are constrained to the range $[0, 1]$, the resulting filter will be stable if the linear part is stable. It is no wonder, therefore, that RBF-based nonlinear adaptive filters are already in use in a number of applications

[11, 12]. As will be shown in this thesis, the RBF is also amenable to straightforward engineering design, so that an RBF-based nonlinear filter may be designed for a wide spectrum of operating requirements.

1.6 Template Matching : An Important Concept

One of the central ideas of pattern recognition is that of *template matching*. Quoting K.S. Fu, once more

> "An intuitively appealing approach for pattern recognition is the approach of 'template-matching'. In this case, a set of templates or prototypes, one for each pattern class, is stored in the machine. The input pattern (with unknown classification) is compared with the template of each class, and the classification is based on a preselected matching criterion or similarity criterion. In other words, if the input pattern matches the template of the i^{th} pattern class better than it matches any other templates, then the input is classified as from the i^{th} pattern class."

Template matching surfaces in various garbs for different data representations in practical PR implementations. In this thesis, it is shown that an RBF net can be interpreted as a two-level template matching operation, so that a substantial part of the design of an RBF net can be reduced to forming templates of different kinds. The first level of template-matching is accomplished by the Gaussian functions in the RBF net :

$$\begin{aligned} \phi(d_i) &= e^{-kd_i^2} \\ where \quad d_i &= distance(sample, prototype_i) \ . \end{aligned}$$

It is evident that the Gaussian function implements a distance-based similarity measure. In the case where the inputs belong to a measurement space in R^n, the templates (or prototypes) are frequently vectors in R^n. With regard to speech recognition, which is a subproblem tackled in this thesis, the basic design aspect developed is how to make speech templates for this level of the RBF that are invariant to time and frequency variations.

The second level of templates occur embedded in the subsequent linear map in the RBF. Each element of the output class vector $\boldsymbol{y}$ is evolved as

$$y_i = < \boldsymbol{w}_i, \boldsymbol{z} >, \quad i = 1, \ldots, m$$

where $\boldsymbol{z}$ is the intermediate vector formed by stacking the outputs of the Gaussian basis functions. The inner product implements a correlation-based similarity measure between $\boldsymbol{z}$ and the prototype weight vector $\boldsymbol{w}_i$. As part of this thesis, this interpretation is used significantly in modularising the linear part of the RBF.

Template-matching operations are local in the sense that their influence is roughly limited to the close proximity (in the sense of metric distance in the case of distance-based matching using basis functions, and in the sense of angular distance in the case of correlation-based matching) of the prototype vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{w}_j$ respectively. The impact of this is that in such a scheme, the information is stored in pockets in the network, and this is what allows identification of specific constituents of the RBF with specific subfunctions. This also allows clubbing together of certain subparts of the RBF into structurally and functionally (in terms of the domain of action) distinct modules, as shown in chapters 5 and 6.

1.7 Focal Points of the Thesis

The thesis has three focal points as follows :

Selection of Basis Functions : Under this, various techniques for selecting basis functions for use in the RBF net, and their impact on the error-complexity trade-off is studied.

Modularisation of RBF Nets : This topic deals with methods to convert RBF nets into modular ANNs. Modularity is enforced both at the level of the basis functions and the subsequent linear map, leading to a set of possible configurations.

Making Templates for Speech Recognition : Since the speech signal is prone to translations and dilations in time and frequency, a stepwise procedure is

outlined so that templates for use in the RBF network are obtained which are invariant to the above-mentioned variations.

1.8 Layout of the Thesis

The thesis is organised into eight chapters, the first of which is the current one, the introduction. This contains the definition of the problem, the notations used, a mention of the main concepts and focal ideas of the thesis, and finally, the current description of the layout of this thesis.

The second chapter is predominantly a review of the literature on RBF nets. The main fact that is sought to be established in this chapter is that the RBF model can be visualised to be the confluence of ideas in several inter-related fields, and that it can be accordingly interpreted in a variety of ways. The development of RBF (or RBF analogous) models from the angles of approximation, regression and fuzzy decision-making is described in detail. The basic techniques for training RBFs using gradient-descent and cluster-and-pseudoinverse are also discussed; it is argued that the former amounts to iterative minimisation of a global error criterion, while the latter involves model optimisation based on an interpretation of the internal structure of the RBF, and therefore, must be preferred. Choice of the latter training model also allows meaningful design of the RBF net in terms of the objectives mentioned earlier.

The third chapter dwells on the effects of two important preprocessing operations on the output of the basis function layer, viz., thresholding (with a small threshold value) and normalisation, on the performance of the RBF net. It is experimentally shown that both improve error performance. With regard to the former, an explanation is provided from the point of view of enforcing a compact support for basis functions, and from the angle of the bias-variance trade-off. The performance enhancement with normalisation is sought to be explained first as the result of a soft-maximisation operation, and next, as a spin-off of the correlation-based linear template matching operation described earlier. The background material corresponding to the two-level match interpretation and the softening of frequently used

nonlinear functions is also provided in this chapter.

Chapter 4 deals with the error-complexity trade-off. There are two dominant ideas — first that error and complexity minimisation usually work against each other, and second that the problem of basis function selection is equivalent to the problem of variable selection in linear regression. The second idea implies the direct application of available methods in statistics for the purpose of basis selection Corresponding to the case of no penalty on model complexity, the orthogonal least squares technique available in literature is cited. Incorporating the complexity minimising requirement into the design can be performed in two ways. The first approach involves converting the complexity into an error penalty, as in the Akaike Information Criterion, and the second, converting the error into a complexity penalty. For the latter, a new information criterion is proposed which penalises equipartitioning of the regressional sum of squares by basis functions, and substitutes the error sum of squares by an equipartitioned (partial) regressional sum of squares The method is also shown to have interesting extensions in the case where the basis functions can be organised into a hierarchy, such that the parent basis functions can be obtained as a linear combination of the children basis functions.

The next two chapters are on techniques for modularising the RBF net. In the first, the basis functions are used as the vehicle of modularisation. This implies that a structure has to be imposed on the basis functions during the clustering stage. Two such structures are used — the tree structure and the flat grid structure. The former uses the LBG vector-quantisation algorithm proposed by Linde, Buzo and Gray [53], while the latter uses Kohonen's Self-Organising-Map (SOM) [54] for clustering with the respective structural constraints. In the former case, entry into nodes at a certain level in the tree, and in the latter, the nearest neighbour quantiser node are used as the gating functions for the respective modules.

In Chapter 6, modularity is imposed on the RBF net by replacing the linear map into piecewise linear maps. Once again, there are two structures imposed — the tree and the flat grid. The former develops into a regression tree (the 1-dimensional analogue of which is found in literature) while the latter takes the shape of a set of unit-rank matrices which are trained by an SOM-like algorithm; the inner product

(of the intermediate vector z) with the rightmost vector v in the unit rank decomposition

$$W = uv^T$$

serves as the gating function in either case. In both chapters, the hierarchical structures are found to produce slightly inferior performance, which is sought to be explained as a result of the accompanying hard binary splits at each level.

Chapter 7 deals with the question of how the speech recognition problem can be accommodated within the RBF framework. For this, the template interpretation is brought to the forefront, and it is explained that the crux of the problem is in obtaining speech templates that are invariant with respect to translation and dilation in time and frequency. A step-by-step procedure for the conversion of speech signal segments into a sequence of spectral vectors is logically developed, and it is illustrated as to how, first by adopting a perceptual mel scale, and later by computing the cepstral coefficients, the effects of the aforesaid transformations in frequency are nullified.

The rest of the chapter is devoted to the problem of deriving a representation for a sequence of vectors that is relatively invariant to translations and dilations in the time index. The important concept developed in this regard is that of forming distance-based tree representations for waveforms. The idea stems from the effort to identify acoustic events as sets of closely-valued points contiguous in time. The implementation of the tree formation procedure is akin to single-linkage clustering with a constraint on the time index. Once the tree is formed, the sequence of nodes at a particular depth is used to represent the original waveform. The representation derives templates which are vector sequences of a fixed length, and have been found to be relatively invariant to monotonic transformations of the time index. These are subsequently used in speech recognition for distinguishing between six closely related diphones. Although the recognition performance for a one-shot developmental scheme was not comparable with that of existing popular implementations, the ease, elegance, and minimal computational and storage requirements of the model certainly warrant attention for this new model.

The last chapter contains a summary of important conclusions and the scope of

future work on some of the aspects of the research effort described in the thesis.

1.9 Data Sets Used in the Thesis

Two data sets have been used in this thesis :

1 The Deterding Phoneme Database for ANNs (details in Appendix A)

2. The TIMIT Speech Database for ASR (details in Appendix B)

The first has been preferred for experimental verification of RBF-design methods throughout the thesis for the following reasons :

- It is available publicly on the Internet.
- It is derived from real speech data.
- It has been developed primarily to test neural networks.
- The results on several common neural networks are provided along with the data.
- It provides a *tough problem* for neural networks (the recognition rate rarely goes above 55%), thereby allowing sufficient scope for comparison between different nets.

The TIMIT Database has been used in chapter 7 for training and testing the ASR implementation using an RBF net. This large, and easily available speech database has an excellent collection of segmented speech samples from an array of speakers, and is the de-facto standard in ASR. An additional advantage of this database is that it is backed by voluminous documentation.

1.9.1 The Generalisation Issue

Generalisation [13] is a measure of how well an ANN performs on the actual problem once the training is complete. It is usually tested by evaluating the performance of

the network on new data outside the training set; frequently, a test set is used for this purpose. Generalisation is heavily influenced by the number of training data samples, the problem complexity and the network size.

Usually a larger training data set is able to provide more training information to the ANN. In fact, a bound on the generalisation error can be established only when the number of training samples is greater than a parameter called the Vapnik-Chervonenkis Dimension (VCdim), which is roughly a measure of the partitioning capability of a system. The VCdim of a system is defined as the size of the largest set S of data samples for which the system can implement all possible $2^{|S|}$ dichotomies on S, where $|S|$ is used to denote the cardinality of S. The rule-of-thumb for training data size selection [13] is that the number of training examples should be approximately ten times the VCdim. For the RBF, VCdim is bounded by [14]

$$\text{VCdim} \leq 2N_W \ln(eN_N) \tag{1.5}$$

where N_W is the number of weights in the network, and N_N is the total number of basis functions in the network. As per the input and output space configurations in this thesis, the VCdim of a network designed with N_N basis functions and having N_O ouput classes, will have

$$\text{VCdim} \leq 2N_O N_N \ln(eN_N) \ . \tag{1.6}$$

Substituting values for the Deterding Vowel Data Set (details in Appendix A), it is found that the training data set is adequate for sufficiently generalising at most a one-node net, going by the rule-of-thumb stated earlier. This could be an important reason for the relatively poor performances noted for various neural networks using the same data set by different researchers (Appendix A) as well as in this thesis.

Of course, since the VCdim formulation largely relies on each of the ANN nodes producing binary outputs, it is difficult to say how well it might be applied to the case of RBFs, where the continuity of the basis functions is one of the basic requirements. However, it is believed that this aberration would only serve to increase the value of the actual VCdim for the RBF.

Chapter 2

Radial Basis Function Nets

In the course of the development of the subject of ANNs, RBF nets have fostered a rather low-key existence. Of the static ANN models today (i.e. those networks whose outputs are functions only of the current inputs and not of the past or future inputs or outputs), the Multilayer Perceptron (MLP) is the most widely used. Yet, by the time the MLP had gained popularity with the publication of the celebrated Back-Propagation algorithm in 'Parallel and Distributed Processing' in 1986 [24], RBFs were already an established field; in 1985, in fact, Powell [17] published a survey of the developments in the subject. A possible reason for the relative obscurity of the RBF compared to the MLP is that it came about as a result of concurrent research by groups working with completely different motivations. It is little wonder, therefore, that these networks occur in the available literature with minor variations under a variety of names, and are amenable to a whole spectrum of interpretations.

This chapter provides a review of the literature on RBF nets. The main fact that is sought to be established here is that the RBF model can be visualised to be the confluence of ideas in several inter-related fields, and that it can be accordingly interpreted in a variety of ways. In this relation, the development of RBF (or RBF analogous) models[1] from the angles of approximation, regression and fuzzy decision-making is described in detail. Then, some of the paradigmatic differences

[1] In each of the formulations, a binary classification problem will be assumed, so that the output will be a scalar $\in \mathbb{R}$; this is in order to maintain correspondence with the cited literature. Extensions to the requirement of a vector output are, however, straightforward in most cases.

between MLPs and RBFs are dwelt upon briefly. Finally, the basic techniques for training RBFs using gradient-descent and cluster-and-pseudoinverse are discussed, and arguments placed in favour of a choice of the latter.

2.1 The Interpolation-Approximation Angle

One of the earliest formulations of the problem of learning an input-output mapping F from a set of examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_K, y_K)\}$ was to regard the same as synthesising an approximation of a multidimensional function. This is equivalent to the reconstruction of a surface from sparse data points A widely-quoted paper by Powell [17] addresses this paradigm, beginning with the noise-free case (i.e. the surface passes exactly through the specified points) of interpolation. He proposes a realisation of the desired map in the form

$$F(\boldsymbol{x}) = \sum_{j=1}^{K} \lambda_j \phi(||\boldsymbol{x} - \boldsymbol{x}_j||) \ , \quad \boldsymbol{x} \in \mathsf{R}^n \ . \tag{2.1}$$

In other words, the interpolation is implemented as the scaled summation of the set of functions generated by ϕ centred at the various data points. The interpolation requirement

$$F(\boldsymbol{x}_i) = \sum_{j=1}^{K} \lambda_j \phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||) \ , \quad i = 1, \ldots, K \tag{2.2}$$

will be satisfied provided that $\boldsymbol{A}$ is nonsingular, where

$$\boldsymbol{A}_{ij} = \phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||) \ , \quad i, j = 1, \ldots, K \ . \tag{2.3}$$

This requirement, however, critically depends on the choice of the function ϕ. Micchelli [18] derived a set of restrictions on the choice of ϕ under which $\boldsymbol{A}$ would be nonsingular[2] as long as the data points were different; *these provide the admissibility conditions for RBFs.* Some of the functions which were shown to satisfy the above-mentioned constraints are listed below :

$$\phi(r) \ = \ \exp[-(\frac{r}{c})^2] \quad (Gaussian) \tag{2.4}$$

[2]It is to be noted that the requirement of locality is not included among these conditions, although the same will be assumed throughout the thesis, and will be crucial for most of the concepts and constructs developed herein.

$$\phi(r) = \frac{1}{(c^2 + r^2)^\alpha}, \quad \alpha > 0 \tag{2.5}$$

$$\phi(r) = (c^2 + r^2)^\beta, \quad 0 < \beta < 1 \tag{2.6}$$

$$\phi(r) = r \quad (linear). \tag{2.7}$$

Poggio and Girosi [19] tried to reformulate the approximation problem in terms of *regularisation* theory. In this approach the desired approximation to the input-output map is obtained through the minimisation of the functional

$$H(F) = \sum_{i=1}^{K} (y_i - F(\boldsymbol{x}_i))^2 + \lambda ||PF||^2 \tag{2.8}$$

where P is a constraint operator (usually a differential operator), $||\cdot||$ is a norm on the function space to which PF belongs, and λ is a positive real number called the *regularisation parameter*. The rightmost term in the above equation can be interpreted as a penalty of the degree of lack of smoothness of the fitted surface. The solution to this equation is shown to be of the form [19]

$$F(\boldsymbol{x}) = \frac{1}{\lambda} \sum_{i=1}^{K} (y_i - F(\boldsymbol{x}_i)) G(\boldsymbol{x}; \boldsymbol{x}_i) \tag{2.9}$$

$$= \sum_{i=1}^{K} w_i G(\boldsymbol{x}; \boldsymbol{x}_i) \tag{2.10}$$

where G is the Green's function of the differential operator $P\hat{P}$, $\hat{P}$ being the adjoint of the differential operator P, i.e.

$$\hat{P}PG(\boldsymbol{x}; \boldsymbol{z}) = \delta(\boldsymbol{x} - \boldsymbol{z}) \ . \tag{2.11}$$

The main impact of the regularisation formulation is the creation of a common framework for approximation using basis functions. The choice of the basis functions is largely governed by the selection of the differential operator P; two appropriate choices of P yield respectively RBF nets and thin-plate spline approximators. It might be remarked in passing that Pao's [8] concept of *functional link nets* is a similar generalisation of the RBF net concept; only that it does not restrict itself to strictly local basis functions, but also uses, depending on the problem at hand, global components e.g. quadratic or cubic combinations of the original input variables. In

fact, it is possible to form single-layer approximation nets even out of sigmoidal functions commonly used in MLPs! The formal justification for all techniques that use transformation into an intermediate set of functions for functional approximation is the Cover's Theorem [15] on the separability of patterns. The import of the same, in crude terms, is that a complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space. For all the approximators mentioned above, the outputs of the basis functions constitute the high-dimensionl intermediate space mentioned in Cover's Theorem.

The RBF (approximation) network described in this section uses one radial basis function centred at each training data point; further, the network is optimised to perform on the training set. As a result, the complexity of the network comes to be of the same order as the training data, and the network suffers from poor generalisation capability; in other words, the derived RBF model gains accuracy on the training set at the cost of deviating from the actual characteristic of the data, of which the training set is but a particular instance. In practice, therefore, it is common to prefer an RBF net with a finite basis trained on a training data set and cross-validated on a test data set. This, however, nullifies some of the optimality properties (e.g. relating to best approximation [16]) of the earlier exact framework.

2.2 The Regression Viewpoint

In statistical terms, it may be convenient to realise the desired input-output map as the regression of the dependent variable[3] Y on the independent variable X. This is equivalent to computing the most probable value of Y for each value of X, based on a finite number of noisy measurements of X and the associated values of Y, and can be realised through the minimisation of the mean-square-error

$$\xi = E(y - \hat{y}(\boldsymbol{x}))^2 \tag{2.12}$$

[3]Notation : $\boldsymbol{x} \in \mathrm{R}^n$ is a particular measured value of the random variable X and $y \in \mathrm{R}$ the same for the random variable Y.

where E is the *expectation* operator, and $\hat{y}$ the implemented function. The solution $\hat{y}(x)$ is thus obtained in the form of the (not necessarily linear) regression

$$\begin{aligned} \hat{y}(x) &= E(y|\boldsymbol{x}) \\ &= \int_{-\infty}^{\infty} y f_{Y|X}(y|\boldsymbol{x}) dy \\ &= \frac{\int_{-\infty}^{\infty} y f_{XY}(\boldsymbol{x}, y) dy}{\int_{-\infty}^{\infty} f_{XY}(\boldsymbol{x}, y) dy} \end{aligned} \tag{2.13}$$

$f_{XY}(\boldsymbol{x}, y)$ being the joint continuous probability density function (PDF) of the vector random variable X and the scalar random variable Y. The joint PDF is however, not available in the general case, and will therefore have to be estimated; this turns out to be the crux of the problem. A solution was proposed by Specht [20] using the class of consistent estimators designed by Parzan [22]. Here, the probability estimator $\tilde{f}_{XY}(\boldsymbol{x}, y)$ is based on sample values $\boldsymbol{x}_i$ and y_i of the random variables X and Y, where K is the number of sample observations and n the dimension of the vector variable $\boldsymbol{X}$ as follows :

$$\tilde{f}_{XY}(\boldsymbol{x}, y) = \frac{1}{(2\pi)^{(n+1)/2}\sigma^{(n+1)}K} \sum_{i=1}^{K} \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i)}{2\sigma^2}\right] \exp\left[-\frac{(y-y_i)^2}{2\sigma^2}\right] . \tag{2.14}$$

It was shown by Parzan [22] that the above density estimator is consistent (i.e. asymptotically convergent to the actual density function at all continuous points), provided that $\sigma = \sigma(t)$ is chosen as a decreasing function of t such that

$$\lim_{t\to\infty} \sigma(t) = 0 \tag{2.15}$$

$$\textit{and} \quad \lim_{t\to\infty} t\sigma^n(t) = \infty . \tag{2.16}$$

Substituting the estimate of the joint PDF obtained in equation 2.14 in equation 2.13, and integrating (following a change in the order of integration and summation), we have

$$\tilde{y}(\boldsymbol{x}) = \frac{\sum_{i=1}^{K} y_i \exp\left[-\frac{D_i^2}{2\sigma^2}\right]}{\sum_{i=1}^{K} \exp\left[-\frac{D_i^2}{2\sigma^2}\right]} \tag{2.17}$$

$$\textit{where} \quad D_i^2 = (\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i) . \tag{2.18}$$

The end-product is therefore a network which is almost identical to the one described in the previous section (assuming Gaussian basis functions), with the only difference that the outputs of the basis functions are normalised by the aggregate sum of the same computed over all basis functions

2.3 The Fuzzy Logic Connection

The local basis functions described in the earlier sections perform a soft[4] partitioning of the input space, and this is of paramount importance in classification applications. In crude terms, this allows different decisions to be taken when the input lies in corresponding localities in the input space. One may tend to find similarities between this formulation and the formal procedure of inferential reasoning using Horn clauses [13] which are of the form

$$\begin{aligned} If\ & \{Object\ has\ property\ A_1\} < and > \ldots \\ & \ldots < and > \{Object\ has\ property\ A_p\} \\ & < then > \{Object\ belongs\ to\ class\ C_j\} \end{aligned} \tag{2.19}$$

where the variables on the left and the right hand side take on truth values in {0,1}, if each property is assumed to be associated with a particular region of the input space.

Fuzzy logic follows as an extension of Boolean logic where the truth value of propositions (and the characteristic functions of sets — modified into *membership functions*) can assume a value continuously over the range $[0, 1]$. Naturally, fuzzy rule-based systems require an additional set of tools over and above those required by their conventional counterparts. First, the antecedents and the consequents as in equation 2.19 have to be *fuzzified* and *defuzzified* respectively. Since the fuzzy sets whose memberships are input to the fuzzy inferencing system are locales on input variables, fuzzification, i.e. converison of crisp numerical input values into fuzzy set memberships involves an operation similar to evolving the outputs of a set of unidimensional basis functions $\mu_{A_j^i}(\boldsymbol{x})$, $i = 1, \ldots, n$, $j = 1, \ldots, p$. Fuzzy intersection

[4] implying that some of the subsets comprising the partition have minor overlaps

(i.e logical *and* function) might be implemented using the product operator so that each fuzzy antecedent will have the truth value

$$\eta_j(\boldsymbol{x}) = \prod_{i=1}^{n} \mu_{A_j^i}(\boldsymbol{x}) \ . \tag{2.20}$$

In the case that the unidimensional membership functions are chosen to be Gaussian, the above form takes the shape of a multivariate Gaussian.

The inferential mechanism in the binary logic case would use a logical intersection of the truth value of the rule and that of the antecedents. Once again, this takes the shape of a product of the truth values of the antecedent and that of the proposition. Since there are several rules, some of them having partial or mutually inconsistent truth values, a mechanism is required so that the outputs of various rules can be integrated to produce a continuous output. This mechanism is called defuzzification, and is usually implemented using the *centroid* method; which, combined with the inferential mechanism outlined above, produces the output

$$\tilde{y} = \frac{\sum_{j=1}^{p} \eta_j(\boldsymbol{x}) w_j}{\sum_{j=1}^{p} \eta_j(\boldsymbol{x})} \ . \tag{2.21}$$

It is therefore observed that the form of the function evolved from the above fuzzy logic inferential methodology — known as the *Fuzzy Basis Function* (FBF) in available literature conforms to the RBF specification. It has the output normalisation feature encountered in the previous section. The restriction on the number of basis functions (i.e that there must be as many of them as the number of input-output data pairs in the training set) is, however, relaxed.

The most important conclusion obtained through being able to derive RBFs through the fuzzy standpoint is that RBFs are amenable to systematic logical interpretation, and that they allow incorporation of linguistic or symbolic information in their structuring.

The above formulation is largely due to Wang, Mendel and Kim [40, 23]; in a similar attempt, Kosko [10] elucidates how the map

$$L : \mathsf{I}^n \mapsto \mathsf{I} \ , \qquad \mathsf{I} = [0, 1]$$

might be regarded as a fuzzy associative memory.

2.4 The MLP and the RBF

In the course of the development of ANNs, the MLP has probably been the most popular feedforward net. The author, however, fosters the belief that the RBF bears structural advantages over the MLP which render it more suitable for design purposes. This section will be devoted to providing some intuitive rationalisation for the same, after a brief description of the history and the structural essentials of the MLP.

2.4.1 The MLP : a Brief Overview

One of the earliest concepts in the history of ANN pattern classifiers was the linear perceptron [25], a generalisation of the threshold logic gate. This was implemented as

$$y = g(z) \tag{2.22}$$

$$where \quad z = \boldsymbol{w}^T \boldsymbol{x} \tag{2.23}$$

$$and \quad g(a) = \begin{cases} 1 & \text{if } a \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

$\boldsymbol{x}$ and y being the input measurement and the output class (assuming binary classification) respectively (other variables $\boldsymbol{w}$ and θ being model parameters), and worked on the basic principle of drawing a hyperplane through the feature space in order to classify patterns. Research in the field plummeted in the late sixties following the publication of the celebrated book on the subject by Minsky and Papert [26] which showed that the perceptron was not capable of separating non-convex regions in the feature space.

The resurgence of the field in the late eighties came with the realisation that by making smooth approximations of the hard threshold function (e.g. the sigmoidal function in Figure 2.1) in a linear perceptron and by increasing the number of layers in the network (one layer feeding its outputs to the next) it might be possible to separate non-convex regions of the input (feature) space from one another by appropriately labelling convex subregions. The MLP thus evolved could be trained

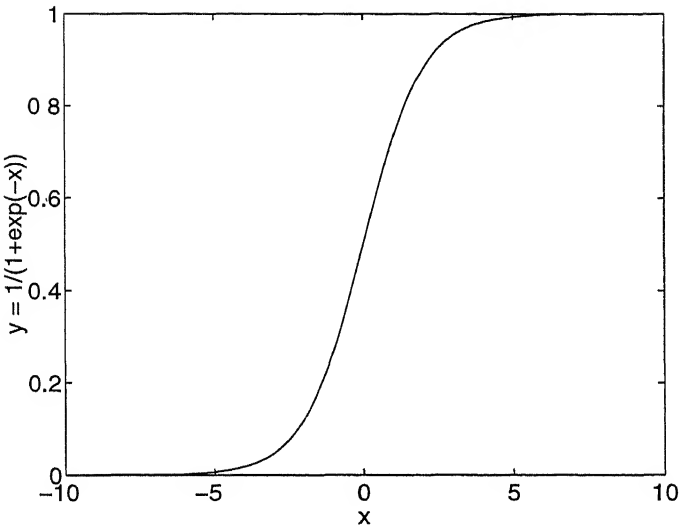

Figure 2 1. The sigmoidal function.

by gradient-descent optimisation techniques, which included the celebrated back-propagation algorithm, and were shown to be *universal approximators*. An important feature of the MLP is global data storage — a characteristic that results from the fact that each component neuron implements a hyperplane, which is a global function (its effect is not limited to a specific region of the input space); as a result, perturbing each neuron usually has the effect of modifying the overall classification operation over multiple, possibly noncontiguous regions of the measurement space.

2.4.2 The MLP and the RBF : a topical comparison

Comparing the structure of the MLP to that of the RBF, it is found that while the MLP has a multilayered (> 2) architecture, the RBF has a flat layout. Both are universal approximators, and have distributed data representations. The respective global and the local data structuring, however, leads to important differences between the two models. The MLP is considered to be a black-box structure, since it is difficult to trace the scope of activity of any individual neuron, and hence the minimisation of a global error criterion through gradient-descent or probabilistic search techniques (e.g. genetic algorithms) provides the only available training method. An RBF, on the other hand, can be trained using the above methods as well as by

others that decompose the training process into a clustering operation and a linear inverse computation operation. The latter utilise the interpretation of the internal structure of the RBF — arguing, for example that the basis functions correspond to data clusters in the input space. These are computationally inexpensive in relation to the former, so the training process of an MLP is frequently more laborious. This reflects particularly during possible retraining, since, for an RBF, the updating has to be performed over a small subset of nodes, and for the linear map, whereas for an MLP, a complete retraining of the entire network is required.

Finally, the interpretability of the RBF network paves the way for structuring the data inside the network — modularising certain parts, and augmenting some others with apriori information (drawing upon the implications of the close relation between RBFs and the FBFs described in the previous section) as dictated by the physical requirements of the PR problem at hand. The internal modelling of the RBF also allows various design trade-offs — e.g. the error-complexity trade-off described in this thesis, so that the RBF can be suited to accommodate design specifications over an entire range. It is for the above reasons that the RBF has been deemed more suitable for design purposes compared to the MLP.

2.5 Main Strategies for Training RBFs

Let us precisely define the form of an RBF net as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{W}\boldsymbol{z} \qquad (2.25) \\ \textit{where } \boldsymbol{z} &= \begin{bmatrix} z_1 \\ \vdots \\ z_p \end{bmatrix} \\ \textit{and } z_i &= \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{\mu_i})^T(\boldsymbol{x}-\boldsymbol{\mu_i})}{\sigma_i^2}\right] \end{aligned}$$

$\boldsymbol{x} \in \mathsf{R}^n$ being the input, $\boldsymbol{y} \in \mathsf{R}^m$ the output, $\boldsymbol{z} \in \mathsf{R}^p$ the intermediate (basis layer output) vector, $\boldsymbol{W}$ an $m \times n$ matrix, and $\mu_i \in \mathsf{R}^n$, $\sigma_i \in \mathsf{R}$, $i \in \{1, \ldots, p\}$ respectively the centres and the variances of the p basis functions used in the model.

There are two main methods of training RBFs. The first implements RBF training as a clustering operation on the input measurement space succeeded by a linear inverse computation, so that the number of basis functions obtained is determined as an outcome of the former. The second assumes an apriori fixed number of basis functions, with the training procedure taking the shape of a process for updating the parameters of the network based on gradient-descent. In the subsections to follow, these methods are treated in a more detailed fashion.

2.5.1 The Clustering-based RBF Training Method

The dominant method of training RBFs from a training data set consisting of K input-output pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ .., (\boldsymbol{x}_K, \boldsymbol{y}_K)\}$ is through the evolution of clusters in the input space followed by an estimation of the linear map.

For the clustering operation, the vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$ are fed into a k-means, Isodata or fuzzy Isodata [27, 28, 8] clustering algorithm to evolve p clusters with centres at $\boldsymbol{\mu}_1, \ \ldots, \ \boldsymbol{\mu}_p$ respectively. In case the above algorithms are not allowed to *coarsen* or *refine* clusters, the number of clusters p has to be specified apriori, and remains fixed throughout. The variances are computed from the data points allotted (wholly or partially according to whether the clustering process is crisp or fuzzy) to the respective clusters.

Each cluster evolved in the clustering process is identified with a radial basis function whose mean is the centroid of the cluster and the variance is the variance of the cluster[5] computed over the set of data values allotted to it. If p such clusters evolve, the linear map is assumed to be implemented through an $m \times n$ matrix, which is subsequently estimated through a generalised inverse or an iterative scheme. The following two subsubsections describe, respectively, the K-means and the Isodata clustering algorithms in algorithmic form. The final subsubsection in the current subsection provides a detailed treatment of the mathematics associated with the estimation of the linear map succeeding the basis function layer in the RBF.

[5]Frequently, for operational purposes, all the clusters are assumed to have the same variance, obtained through averaging over all clusters.

The K-Means Algorithm

- Use the first K data points to initialise the K cluster centres.
- For every subsequent data point
  1. Find the cluster centre closest to it.
  2. Update the cluster centre through centroid computation.
- Declare the final centre locations as the centres of the clusters.

The ISODATA Algorithm

1. Initialise cluster centres by the K-means algorithm.
2. Until the cluster centres do not become stationary repeat
   - Allot the given data points to the clusters based on least distance considerations.
   - Compute the new centres as the centroids of the points allotted to the older ones.

Estimating the Linear Map

Thus equipped with $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$, $i = 1, \ldots, p$, it is easy to compute the best estimate of the matrix $\boldsymbol{W}$ in the least-squares sense, i.e. by minimising

$$\begin{aligned} E_{LS} &= \sum_{k=1}^{K} (\boldsymbol{y}_k - \boldsymbol{W}\boldsymbol{z}_k)^T (\boldsymbol{y}_k - \boldsymbol{W}\boldsymbol{z}_k) \\ &= ||\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{Z}||_F^2 \end{aligned} \tag{2.26}$$

where $||\cdot||_F$ denotes the Froebenius norm, and the matrices $\boldsymbol{Y} = [\boldsymbol{y}_1 \ldots \boldsymbol{y}_k]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1 \ldots \boldsymbol{z}_k]$.

The solution is obtained in the form [30]

$$\boldsymbol{W}^\star = \boldsymbol{Y}\boldsymbol{Z}^+ \tag{2.27}$$

where $\boldsymbol{Z}^{+} = \boldsymbol{Z}^{T}(\boldsymbol{Z}\boldsymbol{Z}^{T})^{-1}$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{Z}$. It is also appropriate to mention here that $\boldsymbol{W}^{\star}$ can also be written as

$$\begin{aligned} \boldsymbol{W}^{\star} &= \hat{\boldsymbol{R}}_{yz}\hat{\boldsymbol{R}}_{zz}^{-1} \\ \textit{where}\quad \hat{\boldsymbol{R}}_{yz} &= \frac{1}{K}\boldsymbol{Y}\boldsymbol{Z}^{T} \\ &= \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{y}_k\boldsymbol{z}_k^{T} \\ \textit{and}\quad \hat{\boldsymbol{R}}_{zz} &= \frac{1}{K}\boldsymbol{Z}\boldsymbol{Z}^{T} \\ &= \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{z}_k\boldsymbol{z}_k^{T} \end{aligned} \tag{2 28}$$

are the estimates of the cross-correlation and autocorrelation matrices respectively, and hence, under assumptions of stationarity of $\boldsymbol{y}$ and $\boldsymbol{z}$, the solution asymptotically converges to the Wiener filter.

It is important to note that training an RBF net by the above method divides the process in two parts; of these the second is linear and optimal, but the first is neither. Effective enumeration of the RBF centres thus becomes a crucial part of RBF design, and will constitute a significant part of this thesis.

2.5.2 Global Training with Gradient-Descent

The other prominent method for training RBF networks comprises iterative updating of the model parameters based on a steepest descent approach [31] to the minimisation of the squared error E_{LS} (equation 2.26). In this case, the number of nodes in the RBF is fixed prior to the training process. Using equations 2.25, 2.26, we have

$$\frac{\partial E}{\partial \boldsymbol{\mu}_i} = -\frac{4}{\sigma_i^2}\sum_{k=1}^{K}(\boldsymbol{W}\boldsymbol{z}_k)^{T}(\boldsymbol{y}_k - \boldsymbol{W}\boldsymbol{z}_k)(\boldsymbol{x}_k - \boldsymbol{\mu}_i) \tag{2.29}$$

$$\frac{\partial E}{\partial \sigma_i} = -\frac{4}{\sigma_i^3}\sum_{k=1}^{K}(\boldsymbol{W}\boldsymbol{z}_k)^{T}(\boldsymbol{y}_k - \boldsymbol{W}\boldsymbol{z}_k)(\boldsymbol{x}_k - \boldsymbol{\mu}_i)^{T}(\boldsymbol{x}_k - \boldsymbol{\mu}_i) \tag{2.30}$$

$$\frac{\partial E}{\partial \boldsymbol{W}} = -2[\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{Z}]\boldsymbol{Z}^{T}\,. \tag{2.31}$$

Starting from random initial values of all parameters, equations 2.29, 2.30, and 2.31 can be used to update the means and the variances of the Gaussian basis functions and the matrix $\boldsymbol{W}$ using the following iterative procedure :

$$\boldsymbol{\mu}_i(n+1) = \boldsymbol{\mu}_i(n) - \eta \frac{\partial E(n)}{\partial \boldsymbol{\mu}_i(n)} \tag{2.32}$$

$$\boldsymbol{\sigma}_i(n+1) = \boldsymbol{\sigma}_i(n) - \eta \frac{\partial E(n)}{\partial \boldsymbol{\sigma}_i(n)} \tag{2.33}$$

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) - \eta \frac{\partial E(n)}{\partial \boldsymbol{W}(n)} \, . \tag{2.34}$$

The iterative process is terminated when the parameters stabilise, or the residual error falls below a specific threshold.

2.5.3 Comparison of the Two Training Methods

The most significant feature of the training procedure stated in the previous subsection is that it aims to seek a global optimum based on a fixed optimality criterion, with the complete RBF structure specified apriori. However, since the error profile will have multiple minima, it is likely that the process will settle at one of these. The clustering-based training method would also provide a suboptimal solution, since the optimal clustering process is NP-hard. However, the common (suboptimal) clustering methods e.g. K-means and Isodata are *greedy*, i.e. they work by adopting the optimal strategy based on immediate considerations, and thus the solution provided by them is close to the optimal solution — something that is usually not guaranteed by the gradient-based technique. For the linear inverse, on the other hand, iterative convergence to the optimal value can be guaranteed if a suitable step-size is chosen. The gradient-based training method also suffers some other disadvantages as the back-propagation nets described earlier in this chapter, including lengthy training regimes and cumbersome incremental updating.

The additional advantage with the cluster-based technique is dynamic model size adaptation based on the training data, provided those variations of clustering algorithms are used which provide the scope of spawning new clusters or merging

close clusters (called *refining* and *coarsening* respectively[6]). This partially relieves the problem of determining the model size apriori.

The biggest advantage of adopting a clustering-based training scheme is that this method uses an interpretation of the internal structure of the RBF. This allows procedures for the explicit selection of basis functions for incorporation in the RBF, which can usually be guided by various design trade-offs, and can be implemented using greedy algorithms. It is primarily for this reason that RBFs will be trained using this method throughout the thesis unless stated otherwise.

[6]The method of implementing coarsening and refining is quite simple. At every iteration of the clustering procedure, e g. Isodata, any training datum whose distance from each of the cluster centres is more than a refining parameter p_r is declared as a new cluster with a single member, while all clusters whose centres are closer than a coarsening parameter p_c are merged.

Chapter 3

Preprocessing the Basis Function Outputs

In the previous chapter, it was observed that the RBF net can be developed from a number of different approaches. The models evolved were found to have minor variations, mostly in the basis function layer. Before going on further into the design of RBF nets, it would be worthwhile to weigh the benefits accrued out of the structural variations so that some of these might be permanently incorporated into the RBF structure.

For some of the discussed RBF formulations, it was necessary for the number of basis functions to be equal to the number of data values in the training set. This feature has been deemed unfit for incorporation since it creates two problems :

1. The number of basis functions becomes very large for a large training set, requiring increased storage and computational resources.

2. The resulting network suffers from poor generalisation capability; what this means is that the model gains accuracy on the training set at the cost of a corresponding decrease in performance on the test set. In statistical terminology, the cause of the same is an increase in the *bias* of the regressional estimate provided by the RBF model.

Two other interesting preprocessing operations on the outputs of the basis functions are thresholding and normalisation. In this chapter, a detailed study will be

made of the impact of these two operations on the overall RBF performance. In order to develop ideas relating to the same, the concepts of softening hard functions, and of positional and angular matching will also be expanded upon.

3.1 Softening Hard Functions

Most neural networks rely crucially for their functioning on a small set of functions that have immediate physical interpretations. The three most commonly used ones among these are the threshold, the impulse and the maximum function. It is apparent that all three of the above-mentioned functions are nonlinear. What actually hampers the training of networks incorporating these functions is their not being differentiable at certain input values, since most techniques use gradient information in one form or the other. It has, therefore, been deemed worthwhile to construct differentiable approximations of the same that converge under some limiting condition to the desired functions. We shall proceed to expand on the theme for the functions mentioned above.

3.1.1 The Threshold Function

This nonlinear function is associated with the implementation of all the ideas which can be expressed in the form

$$if \quad \{x > \eta\} \quad then \quad \{action\} \tag{3.1}$$

where x is a variable, and η a threshold value; the hard threshold (HT) function $f_{TH}(h)$ is used to compute the truth value of the antecedent $\{h \equiv x - \eta > 0\}$ in equation 3.1. It can be realised as

$$f_{TH}(h) = \begin{cases} 1 & if \quad h > 0 \\ 0.5 & if \quad h = 0 \\ 0 & otherwise\ . \end{cases} \tag{3.2}$$

The differentiable approximation to the HT function is the sigmoid function expressed as

$$f_{TS}(h) = \frac{1}{1 + e^{-kh}}\ , \quad \text{where } k \text{ is a constant.} \tag{3.3}$$

This function has a differential

$$f'_{TS}(h) = \frac{ke^{-kh}}{(1+e^{-kh})^2} \tag{3.4}$$

$$= kf_{TS}(h)\{1 - f_{TS}(h)\} . \tag{3.5}$$

We notice that whereas the original function, as in equation 3.2, was neither continuous nor differentiable at $h = 0$, the approximated function satisfies both requirements.

3.1.2 The Impulse Function

This function finds use as an equality verifier, as in clauses of the form

$$if \quad \{x = \eta\} \quad then \quad \{action\} . \tag{3.6}$$

The hard impulse (HI) function $f_{IH}(h)$ can be used to obtain the truth value of the antecedent $\{h \equiv x - \eta = 0\}$ in equation 3.6. This assumes the form

$$f_{IH}(h) = \begin{cases} 1 & if \quad h = 0 \\ 0 & otherwise . \end{cases} \tag{3.7}$$

The function realised in equation 3.7 is both discontinuous and non-differentiable at $h = 0$. Hence the function expressed below can be used to approximate the same.

$$f_{IS}(h) = \exp[-kh^2] \tag{3.8}$$

which is a non-normalised Gaussian. Taking the differential with respect to h,

$$f'_{IS}(h) = -2khe^{-kh^2} \tag{3.9}$$

$$= -2khf(h) . \tag{3.10}$$

The approximation is, therefore, found to be differentiable as well as continuous at $h = 0$.

3.1.3 The Maximum Function

This is a generalisation of the threshold function. Supposing N elements x_i are stacked together to constitute a vector $\boldsymbol{x}$, then the maximum function can be used to implement statements of the form

$$if \quad \{x_i \ is \ the \ winner\} \quad then \quad \{action\} \ . \tag{3.11}$$

This is because of the fact that winning usually involves obtaining a maximum value of a criterion function. The mathematical realisation $f_{MH_i}(x)$ for obtaining the truth value of the antecedent in equation 3 11, stating that $x_i > x_j \ \forall j \neq i$, becomes

$$f_{MH_i}(\boldsymbol{x}) = \begin{cases} 1 & if \ \ x_i > x_j \ \ \forall j \neq i \\ 0 & elsewhere. \end{cases} \tag{3.12}$$

Clearly, the threshold function described earlier is equivalent to taking the maximum of two variables, one of which is 0. The maximum function is discontinuous as well as non-differentiable at numerous points. The differentiable approximation to the same is

$$f_{MS_i}(\boldsymbol{x}) = \frac{e^{kx_i}}{\sum_{j=1}^{N} e^{kx_j}} \ , \quad where \ k \ is \ a \ constant. \tag{3 13}$$

The differentials of $f_{MS_i}(\boldsymbol{x})$ are as follows :

$$\frac{\partial f_{MS_i}(\boldsymbol{x})}{\partial x_i} = ke^{kx_i}\left[1 - \frac{e^{kx_i}}{\sum_{j=1}^{N} e^{kx_j}}\right] \tag{3.14}$$

$$= ke^{kx_i}[1 - f_{MS_i}(\boldsymbol{x})] \tag{3.15}$$

$$and \ \ \frac{\partial f_{MS_i}(\boldsymbol{x})}{\partial x_j} = -\frac{ke^{(kx_i + x_j)}}{\sum_{l=1}^{N} e^{kx_l}} \ \ for j \neq i \tag{3.16}$$

$$= ke^{kx_j} f_{MS_i}(\boldsymbol{x}) \ . \tag{3.17}$$

3.2 Activation-Based Selection of Basis Functions using Thresholding

If we look at the expression

$$\boldsymbol{y} = \sum_{i=1}^{N} \boldsymbol{w}_i \phi_i(\boldsymbol{x}) \tag{3.18}$$

the term $w_i\phi_i(x)$ represents[1] the contribution of the i^{th} basis functions, while the term $\phi_i(x)$ in isolation is its activation. In a sense it is the measure of how well the test vector x matches the template μ_i If the number of possible basis function is large to begin with, it is possible that a number of them will contain redundant information, further, nodes that have small activation would substantially contribute to the model bias. A method of preprocessing the outputs of the basis functions would be, therefore, to select those on an occasion, the activations of which are above a certain threshold t_a, i.e.

$$y = \sum_{i=1}^{M} w_i\phi_i(x) \tag{3.19}$$

where $M \leq N$ and $\phi_i(x) \geq t_a$, $\forall i \in \{1, \ldots, M\}$. In case of an RBF implementation using Gaussian basis functions, the implication of activation-thresholding is in terms of their truncation, thereby limiting their support, i.e.

$$\phi_i(x) = \begin{cases} \exp\left(-\frac{||x-\mu_i||^2}{\sigma_i^2}\right) & if\ ||x-\mu_i||^2 \leq -\sigma_i^2 \ln(t_a) \\ 0 & otherwise. \end{cases} \tag{3.20}$$

The resulting truncated basis functions would appear as in Figure 3.1.

3.2.1 Experimental Results

The thresholding scheme was tried out on RBF implementations using the Deterding phoneme data set (Appendix A) using 11, 18 and 29 clusters respectively. Thresholding of the basis function outputs was carried out both at the training and the testing phases. Figure 3.2 depicts the way the performance was found to vary with the threshold, and Figure 3 3 the corresponding average number of active basis functions used in the PR process.

From Figure 3.2 it is observed that starting with a given initial number of clusters, the performance initially increases with an increase in the number of basis functions, peaks and finally decays. This can be explained in terms of a bias-variance trade-off [32]. When the number of active basis functions goes down as the result of an

[1] $\phi_i(x) = \phi(||x - \mu_i||)$ in equation 1.1.

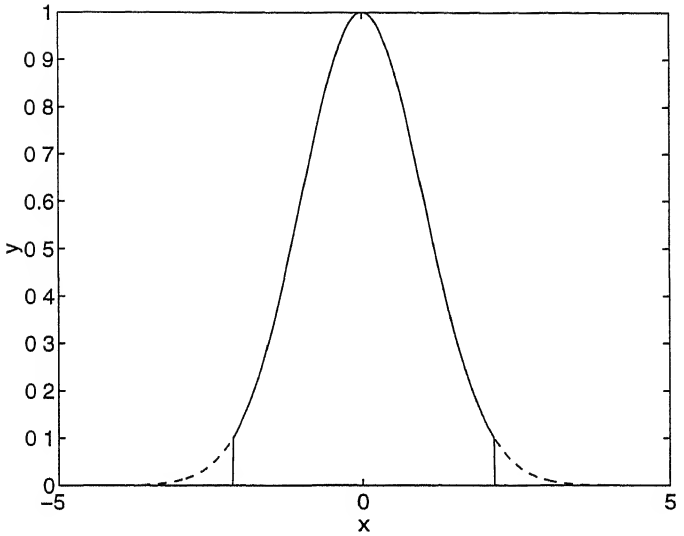

Figure 3 1: The truncated Gaussian function produced with a threshold of 0.1 The dashed line shows the original curve.

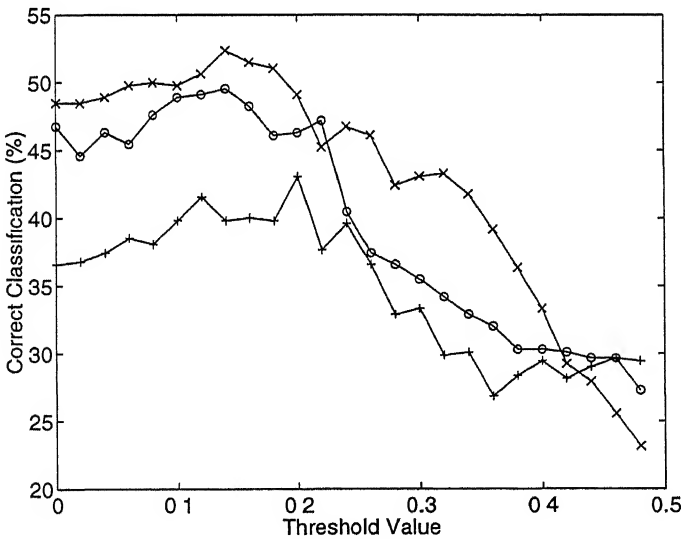

Figure 3.2: Variation of RBF performance with respect to the output threshold value. The '×', 'o' and '+' signs indicate the curves corresponding to 29, 18 and 11 initial clusters respectively.

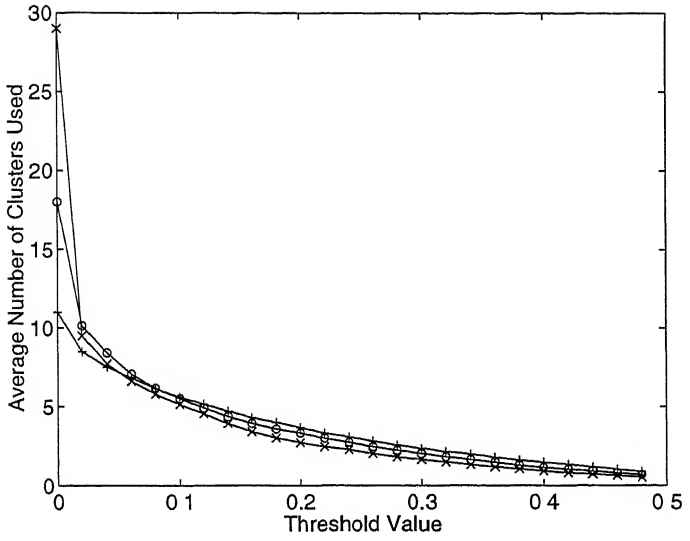

Figure 3.3· Variation of the average number of basis functions used by the RBF with respect to the output threshold value t_a. The '×', 'o' and '+' signs indicate the curves corresponding to 29, 11 and 18 initial clusters respectively

increase in the threshold t_a, the bias of the regression increases, and the variance decreases; in other words, the approximating power of the network declines, but the available data becomes more and more adequate for training the network.

It is also noted by comparing the three different curves in the same figure that the performance is better for a larger initial number of clusters although the average number of clusters roughly coincides in all cases for a threshold value above 0.04 (Figure 3.3). This can be explained in terms of an increase in the network's approximation capacity, with part of the effect of increasing variance being nullified by the thresholding operation.

3.2.2 Estimating the Threshold

In the previous section, the value of the threshold was varied exhaustively over a wide range; for a large problem, the process would be computationally expensive. A straightforward solution may, however, be suggested in the form of a gradient-based solution (coupled with the training of the linear part). A problem faced at this juncture is that the truncation of the basis functions will reduce the smoothness of

the aggregate function $F(\boldsymbol{x})$, and may make training by gradient-descent untenable. There is, however, a way out if we represent ϕ_i as

$$\begin{aligned} \phi_i(\boldsymbol{x}) &= g_i(\boldsymbol{x})u_i(\boldsymbol{x}) \\ where\ u_i(\boldsymbol{x}) &= \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{\mu}_i||^2}{\sigma_i^2}\right) \\ and\ g_i(\boldsymbol{x}) &= \begin{cases} 1 & if\ ||\boldsymbol{x}-\boldsymbol{\mu}_i||^2 + \sigma_i^2 \ln(t_a) \leq 0 \\ 0 & otherwise. \end{cases} \end{aligned} \tag{3.21}$$

It is possible to approximate $g_i(\boldsymbol{x})$ using the nonlinear function

$$h_i(\boldsymbol{x}) = \frac{1}{1+\exp[k_c(||\boldsymbol{x}-\boldsymbol{\mu}_i||^2 + \sigma_i^2 \ln(t_a))]} \tag{3.22}$$

where k_c is a positive constant. The resulting function

$$\tilde{\phi}_i(\boldsymbol{x}) = h_i(\boldsymbol{x})u_i(\boldsymbol{x}) \tag{3.23}$$

is both continuous and differentiable; once again, it does not have a compact support in the strict sense as a penalty for *softening* the hard threshold, but its decay to zero at the prescribed boundaries is faster than for $u_i(\boldsymbol{x})$ at the same points.

At this juncture, there might be a temptation to estimate the parameter t_a adaptively while training for the weights of the basis functions $\boldsymbol{w}_i$, starting with the initial value of $t_a = \alpha$, a small positive constant. Again, assuming that

$$\begin{aligned} \tilde{\boldsymbol{y}} &= \sum_{i=1}^{N} \boldsymbol{w}_i h_i(\boldsymbol{x})u_i(\boldsymbol{x}) \\ \boldsymbol{e} &= \boldsymbol{d} - \tilde{\boldsymbol{y}} \end{aligned}$$

is the instantaneous error, $\boldsymbol{d}$ being the expected output, and that

$$\mathcal{E} = \sum_{k=1}^{K} \boldsymbol{e}^T[k]\boldsymbol{e}[k] \tag{3.24}$$

is the sum-of-squares error for K training points, we can easily derive

$$\frac{\partial \mathcal{E}}{\partial t_a} = -\frac{2k_c}{t_a} \sum_{i=1}^{N} \sigma_i^2 \sum_{k=1}^{K} (\boldsymbol{e}^T[k]\tilde{\boldsymbol{y}}[k])\{1 - h_i(\boldsymbol{x}[k])\} \tag{3.25}$$

and substitute the same in the equation

$$t_a[n+1] = t_a[n] - \eta_{t_a} \frac{\partial \mathcal{E}}{\partial t_a} \tag{3.26}$$

where η_{t_a} is a suitable constant, and n denotes the iteration number.

3.3 A Note on Local Distributed Processing

Traditional ANN systems like the MLP have often cited *globality*[2] of information storage as a virtue; as its beneficial spin-offs have been cited increased robustness of decisions and graceful degradation of performance with failure proliferation. However, such a system, with increase in complexity, creates a formidable training problem in terms of seeking out a global optimum from a multitude of local optima, and placing a requirement of global retraining on being faced with new data inputs. Further, such a system cannot provide insights into the way the data is internally organised. In this thesis, a primary contention that will be put forth is that the advantages of the so-called *globality* are largely those accruing from a distributed processing mechanism. The distinction between global and distributed processing[3] arises from the possibility of uncoupling of data elements scattered throughout the network in the latter. As a result, it might be possible to construct a distributed processing system through a level-by-level integration of modules without losing any of the advantages of the earlier system; where the modules themselves, viewed at the proper resolution at which they are integrated, utilise information that is **local** to them.

This is possible, for example, if the modules implement some kind of a template-matching operation; the templates attached to the module provide the necessary local information. For the RBF, the operation of the basis functions and the linear map can each be shown to be equivalent to different kinds of template-matching as explained in the following subsections.

3.3.1 Template Matching using Distance

The basis functions in the RBF are of the form

$$\phi_i(\boldsymbol{x}) = \phi(d(\boldsymbol{x}, \boldsymbol{x}_i)) \ , \qquad \boldsymbol{x}, \boldsymbol{x}_i \in \mathsf{R}^n \tag{3.27}$$

[2]Here globality is in the sense that the information is indistinguishably dispersed throughout the network, and it is only meaningful to talk about the network as a whole

[3]the former is a subclass of the latter

where $d(\boldsymbol{x}, \boldsymbol{x}_i)$ is a distance criterion, commonly the L_2 norm of $(\boldsymbol{x} - \boldsymbol{x}_i)$, and $\phi \in L_2(\mathrm{R})$ is the non-normalised Gaussian function

$$\phi(x) = e^{(-k_P x^2)} \tag{3.28}$$

which also satisfies the requirement that

$$\xi(\phi) = \int_{-\infty}^{\infty} \frac{|\phi(x)|^2}{|x|} dx < \infty \; . \tag{3.29}$$

Equation 3.27 explicitly portrays the dependence of the basis function outputs on the distance from the template vector $\boldsymbol{x}_i$, while equation 3.29 imposes the functional-analytic locality requirement on ϕ. As can be observed, $d(\boldsymbol{x}, \boldsymbol{x}_i)$ is a dissimilarity criterion which is converted into a similarity measure using ϕ. The aggregate function $\phi_i(\boldsymbol{x})$ is, therefore, *positionally localised* at $\boldsymbol{x}_i$; $\boldsymbol{x}_i$ will be called the positional template (PT), and the mechanism of matching using distance from the PT, positional localisation.

3.3.2 Template Matching using Correlation

The linear part of the RBF generates each component y_j of the output by the operation

$$y_j = < \boldsymbol{w}_j, \boldsymbol{z} > \tag{3.30}$$

adopting the same notations as in equation 1.3. This is tantamount to obtaining the match between $\boldsymbol{w}_j$ and $\boldsymbol{z}$ by computing their inner product. Particularly in the case where both $\boldsymbol{w}_j$ and $\boldsymbol{z}$ are normalised to unit (L_2) norm,

$$y_j = cos(\theta_j) \tag{3.31}$$

where θ_j denotes the angle between $\boldsymbol{w}_j$ and $\boldsymbol{z}$. This has a very interesting relationship with the distance-based match between the two vectors as follows :

$$\begin{aligned} e^{-k_P||\boldsymbol{z}-\boldsymbol{w}_j||^2} &= e^{-2k_P||1-cos(\theta_j)||} \\ &\approx g(cos(\theta_j)) \quad for \;\; k_P \approx 4 \\ &= g_c(\theta_j) \\ where \;\; g(t) &= \begin{cases} 1 & if \;\; t = 0 \\ 0 & otherwise. \end{cases} \end{aligned} \tag{3.32}$$

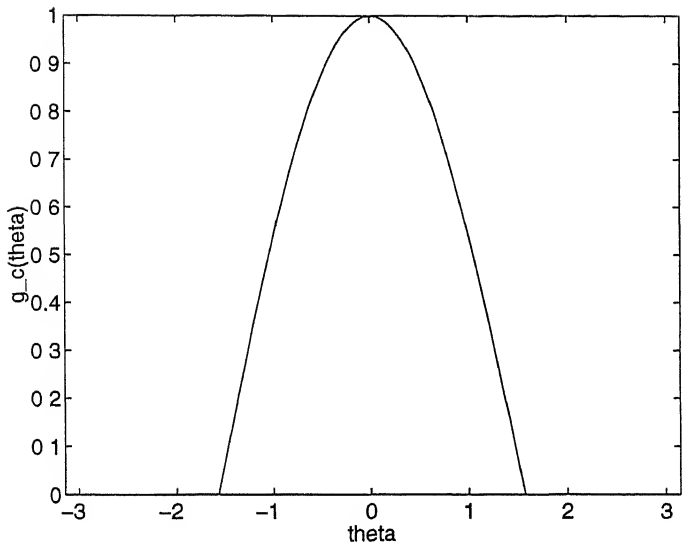

Figure 3.4. Plot of the rectified cosine function $g_c(\theta)$ against θ.

What thus emerges is that under the restrictions of normalised vectors and specified constant k_P, template-based matching using distance degenerates into template-based matching using correlation with the additional stipulation that negative correlations are forced to zero.

The degree of match by correlation $g_c(\theta_j)$ is plotted against θ_j in Figure 3.4. Once again, as θ_j increases, the match decays to zero. The matching operation is, therefore, localised in angle at the orientation of $\boldsymbol{w}_j$; $\boldsymbol{w}_j$ will be referred to as the *orientational template* (OT), and the process of matching by correlation as *orientational localisation.*

3.3.3 Some Thoughts on Hierarchical Levels of Localisation

It has been illustrated in the previous two subsections that the RBF map (with normalised basis function outputs — this will be dealt with later in this chapter) is equivalent to an orientational localisation applied to a prior positional localisation; the interpretation that follows is that the map is constructed by a hierarchical organisation of matching (localisation) operators. It might, therefore, be possible to increase the number of layers of localisation [34], thereby forming multilayer basis function nets, or to replace the existing localisation operators with others.

Looking at the current formulation, however, the choice of the matching operators seems to be appropriate. The selection of the initial PTs defined on the input measurement space is in conformity with the view that a number of points from the same class will form clusters. The localisation thereafter could again have been positional, but certain advantages would have been lost; first that the only available method of supervised training not using global optimisation would probably be Learning Vector Quantisation (LVQ) due to Kohonen [29]; second that the model would fail to be the point of convergence of the diverse motivations as illustrated in the previous chapter; and finally that there would be no way to integrate the soft outputs of the diverse cluster units — in other words, the link from clusters to classes would be severed. It therefore seems inevitable that the hierarchically organised template-matching layers are chosen in the same order as in an RBF.

3.4 Normalising the Basis layer Output Vectors

In sections 2.2 and 2.3, the regression nets and FBF nets respectively were found to have the characteristic that each basis function was normalised as

$$\tilde{\phi}_i(\boldsymbol{x}) = \frac{\phi_i(\boldsymbol{x})}{\sum_i \phi_i(\boldsymbol{x})}\ , \quad where \quad \phi_i(\boldsymbol{x}) = \phi(||\boldsymbol{x} - \boldsymbol{\mu}_i||_2) \tag{3.33}$$

and ϕ is as in section 1.3. In the current section, some intuitive arguments justifying the normalisation operations will be placed, followed by experimental results on the Deterding data set.

3.4.1 Justifications for Basis Layer Output Normalisation

Basis function layer output normalisation has the important effect that the inputs to the RBF net that evoke smaller responses from all basis functions (mostly at the boundaries between classes) have proportional contribution in training the classifier; as a result, it is expected that the operation will enhance the discriminative power of the overall network for PR. This becomes especially important if the outputs of the basis functions are first thresholded, forcing a large number of the same to zero.

Then again, normalisation may be viewed from the biological angle of competition Let us assume that the resource allotted to the basis functions is unity amplitude (or more appropriately, as will be explained shortly, unity energy), and this is distributed in a manner that favours the units with higher original activation. Proportionate distribution dictates that the outputs of the basis functions be normalised by the sum of the activation of all the basis functions.

It is interesting to note that the same solution to the problem of resource distribution may be arrived at by using a different formulation. The strategy still maintains the competition perspective, but decides to concentrate upon the winner rather than on the proportionate distribution of limited resources, using as basis the measure of similarity to templates embedded in the basis functions. Supposing that all the benefits of the outcome for a given input accrue to the basis function with the maximal activation, the linear map for the RBF net will operate on the output of a maximum function operating on a set of variables clubbed into a vector as in section 3.1.3.

If there are p basis functions centred respectively at $\boldsymbol{\mu}_i$, $i = 1, \ldots, p$, the square of the distance of the input $\boldsymbol{x}$ from these is given by

$$d_i^2 = ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2 \ , \ i = 1, \ldots, p \ . \tag{3.34}$$

Now similarity can be looked upon as decreasing with distance, and hence $-d_i^2$ can be looked upon as a measure of similarity of $\boldsymbol{x}$ with the vector $\boldsymbol{\mu}_i$.

Implementing a softmax on this as in equation 3.13 results in individual elements of the vector of the form

$$z_i(\boldsymbol{x}) = \frac{e^{-k||\boldsymbol{x}-\boldsymbol{\mu}_i||^2}}{\sum_{j=1}^{p} e^{-k||\boldsymbol{x}-\boldsymbol{\mu}_j||^2}} \ , \quad \textit{where } k \textit{ is a constant} \tag{3.35}$$

which, when k is interpreted as the spread σ^2, turns out to be the normalised outputs of radial basis functions. Hence, a normalised RBF can be regarded as the implementation of a softmax operation on the similarity measure.

A completely different interpretation evolves, however, if we shift our attention from the basis functions themselves to the subsequent linear map; let us assume that this involves an $m \times p$ matrix $\boldsymbol{W}$ operating on $\tilde{\boldsymbol{z}}$, the non-normalised outputs

of the basis functions, where

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_m^T \end{bmatrix}$$

w_j, $j = 1, \ldots, m$, are p-dimensional coloumn vectors. Then we have

$$y_j = w_j^T \tilde{z} \ . \tag{3.36}$$

As illustrated in section 3.3.2, this can be regarded as OT matching provided $\tilde{z}$ is normalised to have unit norm. Since the latter is an important prerequisite for the former, without normalisation, the interpretation of the RBF as a two-level template-matching operation becomes invalid, and the corresponding modularisation technique developed in chapter 6 ceases to work.

For this, let us consider the family of metrics

$$L_q(\tilde{z}_1, \tilde{z}_2) = \{\sum_{i=1}^{p} |\tilde{z}_{1i} - \tilde{z}_{2i}|^q\}^{1/q} \tag{3.37}$$

where q is an integer. The corresponding set of norms are given by

$$||\tilde{z}||_q = \{\sum_{i=1}^{p} |\tilde{z}_i|^q\}^{1/q} \ . \tag{3.38}$$

It is immediately apparent that normalising $\tilde{z}$ by the sum of its elements as described earlier is equivalent to normalising by

$$||\tilde{z}||_1 = \sum_{i=1}^{p} |\tilde{z}_i| \ . \tag{3.39}$$

There is a catch, however, in that the metric L_1 does not satisfy the parallelogram law [33], and hence cannot be used to derive a corresponding inner product. L_2, on the other hand, is compatible with the inner product

$$< \tilde{z}_1, \tilde{z}_2 >= \tilde{z}_1^T \tilde{z}_2 \ . \tag{3.40}$$

Using the L_2 metric for normalisation in equation 3.35, we have

$$z_i(x) = \frac{e^{-k||x-\mu_i||^2}}{\{\sum_{j=1}^{p} e^{-2k||x-\mu_j||^2}\}^{1/2}} \ . \tag{3.41}$$

It is also interesting to note that the function

$$f(z) = \frac{e^{kx_i}}{(\sum_{i=1}^{p} e^{2kx_i})^{1/2}} \tag{3 42}$$

is also a valid softmax function, and produces values in $[0, 1]$. However, since

$$L_2(\tilde{z}_1, \tilde{z}_2) \leq L_1(\tilde{z}_1, \tilde{z}_2)$$

the values as obtained through the normalisation as in equation 3.42 yields slightly larger values In simulations, the performance of the two cases have been found to differ minimally. For the sake of coherency of formulation, the form in equation 3.42 will be preferred to that in equation 3.35 throughout the rest of the thesis.

3.4.2 Experimental Results

The thresholding scheme implemented upon the Deterding vowel data set in subsection 3.2.1 was augmented using subsequent normalisation of the (thresholded) basis function output vectors using the L_2 norm. Figure 3.5 depicts the way the performance varies with the threshold; the difference between these characteristics and the ones shown in Figure 3.2 arise from the normalisation operation.

Comparing Figure 3.5 and Figure 3.2, it is observed that the rough order and characteristics of the curves remains the same, with an improvement in the recognition performance in the former. The highest recognition performance was 55% achieved in the thresholded and normalised case. The results validate the earlier contention that normalisation of basis function layer outputs would lead to an improvement in the overall performance of the RBF net.

Repeating the above experiment using the L_1 norm for normalisation of the basis function outputs yielded almost identical results (the results with the L_2 norm were marginally superior), and it is concluded that either may be used in an RBF depending on the convenience of the situation.

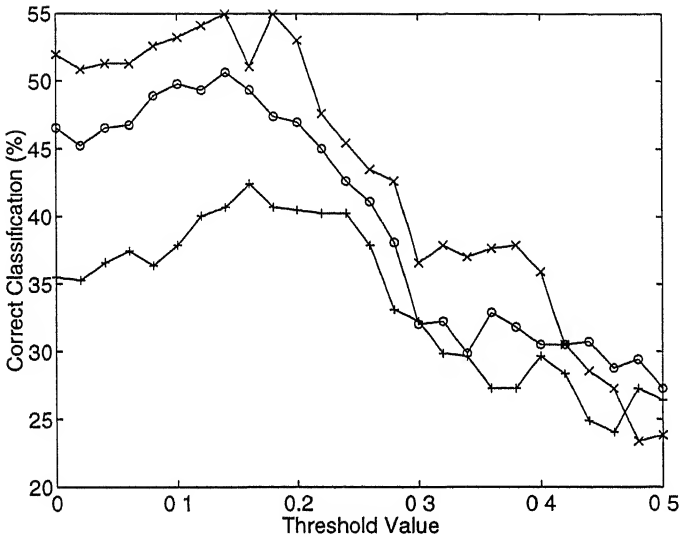

Figure 3.5: Variation of RBF performance with respect to the output threshold value for thresholded and normalised basis function outputs. The '×', 'o' and '+' signs indicate the curves corresponding to 29, 18 and 11 initial clusters respectively.

Chapter 4

Error and Complexity based approaches in Basis selection

In the previous chapter, it has been seen how the simple operation of thresholding the outputs of the basis functions in the RBF net can lead to an effective reduction in the number of basis functions during field operation as well as provide improved performance. The basic premise there was that the number of basis functions forming the complete model of the input-output relationship embodied in the data was far in excess of what might be suitable for proper generalisation on the limited data set. In the general case, however, model error and complexity work against each other. This is because a more complex RBF model usually has a greater approximation power, provided, of course, that a substantial amount of training data is available[1].

In subsection 2.5.1, it was assumed that basis functions could be obtained through clustering the input data; the basis of this strategy was the premise that each basis function was identifiable with a corresponding data cluster. However, there are other direct methods to obtain basis functions without using the cluster interpretation. For the same, we shall assume apriori that a large set of basis functions $\mathcal{B}$ is available to us by some means, from which a subset $\mathcal{B}_1$ needs to be chosen to solve the mapping problem at hand. In his book "Neural Darwinism," Edelman [36] argues

[1]This is required in order to minimise model bias.

that *selection* of functionalities from a pre-existing library of the same encoded in the chromosomes (each cell in the organism inherits an identical set) is the basis for cellular differentiation in living multicellular organisms, and extrapolates the same to the case of early brain formation. The current chapter will draw on similar logic in deriving basis functions for use in the RBF net.

In practical situations, a whole lot of basis functions can usually be generated based on physical knowledge about the problem and through the application of relevant heuristics As an example, just considering the entire family of normalised Gaussian basis functions with varying means and variances, one can find a set $\mathcal{B}$ that is infinitely large to choose from — assuming, that is, it suits the problem formulation.

The strategy that will be adopted in this chapter will be to formulate the basis subset selection problem into an optimisation framework. For this, we shall concentrate on the linear map in the RBF net and treat the basis functions as inputs to the same. The following sections will comprise a more detailed discussion of the guidelines aiding in basis selection, followed by a series of greedy stepwise basis selection algorithms. The techniques which penalise the regression error are explored first. As part of the same, it is shown how the statistical methods for variable selection in linear regression can be suitably utilised for basis selection. The topic then veers into the methods using error-complexity trade-offs for basis function selection. Finally, a new information criterion is proposed for the same purpose, and its extension to the case of hierarchical basis functions is derived.

4.1 Basis Selection : Broaching the Topic

This section will involve a discussion of the suitability of the criteria for basis selection. We shall choose as our starting point the realm of signal decomposition for linear time-invariant systems, a familiar topic in mainstream signal processing. The resulting analysis makes extensive use of the form

$$x(t) = \sum_n c_n \phi_n(t) \tag{4.1}$$

where $x(t)$ is a given signal, and the ϕ_ns are sines, cosines, or complex exponentials. It is not difficult to understand the reason for this choice of basis functions (for signals), since these happen to be the signal-independent eigenfunctions for any linear time-invariant (LTI) system, i.e.

$$L_i\phi_n(t) = A_{in}\phi_n(t) \tag{4.2}$$

where A_i is a complex amplification factor, and hence,

$$L_i x(t) = \sum_n A_{in}c_n\phi_n(t) \ . \tag{4.3}$$

The motivation behind the choice of complex exponentials as basis functions (vis-a-vis the signal at hand) is that they are units that are form-invariant to transformations by LTI systems.

There are, of course, in spite of the apparent similarities in structure, fundamental differences between the above and the basis functions that have been mentioned in the earlier sections in connection with nonlinear system realisation via the RBF architecture. First and foremost, that with the former, the motivation is signal decomposition, while, with the latter, it is system decomposition; second, the presence of a linear time invariant operator domain associated with the former. What the differences imply is that our interest in the form in equation 4.1, assuming that t is the input, and $x(t)$ is the desired output, is not governed by further considerations of invariance as in 4.2 The data representation is the be-all and end-all of the problem. Hence, minimisation of approximation error and reduction in model complexity become the main factors overriding the choice of basis functions. The first is a measure of the accuracy of the representation, while the second is a reflection of the resources consumed in the modelling process. The latter is just as important in engineering design as the former, since the allocated resources (in this case, storage and computational) are limited. With the RBF, as with most other models, performance improves with an increase in the model size (meaning greater number of basis functions) because it allows (roughly) a finer partition of the input space for subsequent class labelling; as a result, error and complexity minimisation work against each other.

The error-complexity trade-off is reflected in the original RBF formulation, working from the standpoint of approximation theory. It may be recalled (section 2.1 that Poggio and Girosi's [19] cost functional was of the form

$$E(F) = E_a(F) + \lambda E_r(F) \tag{4.4}$$

$$= \frac{1}{2}\sum_{i=1}^{N}[d_i - F(x_i)]^2 + \frac{1}{2}\lambda||PF||^2 \tag{4.5}$$

where E_a is the approximation error, and E_r the regularisation term roughly translating into the degree of lack of smoothness of the approximated function, P being a differential operator and λ the regularisation parameter. However, the very fact that the solution to the regularisation problem prescribes the use of all the training data points as centres of basis functions makes it apparent that a premature conversion of the data-fitting problem into an optimisation formulation is hardly an efficient option.

With this backdrop, we shall set out to probe various methods of selection of basis functions. In the following subsection, the strategy of basis selection will be further narrowed down into a class of tractable solutions.

4.1.1 Stepwise Basis Function Selection

It has been mentioned earlier in the current section that the minimisation of regression (roughly misclassification) error and model complexity are the two main issues in basis selection. Further, since these work against each other, the two are frequently incorporated into a single cost function; then the direct method of basis selection involves minimisation of the same.

As before, however, global minimisation of the cost function proves to be an intractable problem. The tractable suboptimal solution lies in selecting the basis functions one-at-a-time. The basis selection process, therefore, comprises a series of steps, each of which is *greedy* in the sense that it is optimal under the given circumstances without ensuring optimality for the entire selection process. In the following sections, a set of greedy stepwise basis function selection procedures will be described, starting with procedures that penalise model error in isolation.

4.2 Basis Selection through Error Minimisation

In this section, two main techniques for basis selection will be described. The first follows a regressional formulation of the error minimisation problem, while the second uses the concept of novelty in order to implement basis function selection.

4.2.1 The Regressional Formulation of the Basis Selection Problem

In this subsection, a solution to the problem of basis function selection is described, using techniques available in statistical regression theory. The topic will be developed by first providing some background material on linear regression, followed by the explanation that the problem of variable selection in linear regression and the current problem are one and the same. Some of the available solutions in statistical literature will be cited; in this regard, mention will be made of the popular Orthogonal Least Squares (OLS) algorithm used for RBFs that has its equivalent in statistical literature.

- *Linear Regression : A Brief Discussion*

The basic ideas of regression relating to the RBF are discussed in section 2.2. The hard part of finding a general solution for a regressional formulation of the PR problem stems from the problem of the estimation of the joint probability density functions involved If our knowledge about the problem, however, dictates a certain form of a map from the input to the output, the same might easily be incorporated into the regressional model under additional assumptions relating to deviations from the ideal map. Supposing, e.g., that the map is linear, it could make sense to model the perturbation as an additive random signal. We therefore have the linear map

$$y = \sum_{i=1}^{p} w_i z_i + e \tag{4.6}$$

where e is uncorrelated residual *error*.

The idea of regression is intimately linked to the idea of minimisation of the mean-square error. For the linear input-output map, this takes the form of

$$\mathcal{E}_{ms} = E[(y - \sum_{i=1}^{p} w_i z_i)^2] \tag{4.7}$$

E being the probabilistic expectation and $\boldsymbol{z} = [z_1\ z_2\ \cdots\ z_p]^T$ and y being the inputs and outputs respectively.

However, in most cases, since the probability density functions are not available apriori, it becomes practical to minimise the residual sum of squares error

$$\mathcal{E}_{ss} = \sum_{k=1}^{K}[y(k) - \sum_{i=1}^{p} w_i z_i(k)]^2 \tag{4.8}$$

The solution obtained for the above can be shown to converge to that applicable to equation 4.7 as $K \to \infty$. Supposing that we constitute

$$\begin{aligned} \boldsymbol{y} &= [y(1)\ y(2)\ \cdots\ y(K)] \\ \boldsymbol{z}_i &= [z_i(1)\ z_i(2)\ \cdots\ z_i(K)] \qquad (4.9) \\ \boldsymbol{w} &= [w_1\ w_2\ \cdots\ w_p] \qquad (4.10) \\ \textit{and}\ \boldsymbol{Z} &= [\boldsymbol{z}_1^T\ \boldsymbol{z}_2^T\ \cdots\ \boldsymbol{z}_K^T]^T \end{aligned}$$

the linear regression formulation evolves into the minimisation of

$$\mathcal{E}_{ss} = (||\boldsymbol{y} - \boldsymbol{w}\boldsymbol{Z}||_2)^2 \tag{4.11}$$

which leads us to the familiar solution

$$\boldsymbol{w}^\star = \boldsymbol{y}\boldsymbol{Z}^+ \ ,\ \ \textit{where}\ \ \boldsymbol{Z}^+ = \boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{Z}^T)^{-1} \ . \tag{4.12}$$

However, this estimate assumes that all the input variables do significantly contribute to the output y; in real situations, a number of components of $\boldsymbol{z}$ may not be suitably correlated with the output y, and may merely be contributing to the model bias. Eliminating these would result in reduction both in the model bias and the computational and storage requirements of the system.

Variable Selection in Linear Regression

There are a set of greedy algorithms in statistical literature [37, 38] to tackle the problem of variable subset selection which operate through the step-by-step choice of variables using local minimisation of the regression sum of squares error measure. The most notable among these is the *Orthogonal Least Squares* (OLS) algorithm [41, 40] which is a combination of the orthogonal reduction and *forward selection* methods developed in [37, 41].

For this, the regression problem is reformulated for a single input variable z_j using the error function

$$\mathcal{E}_{ss}(j) = (||\boldsymbol{y} - w_j \boldsymbol{z}_j||_2)^2 \ . \tag{4.13}$$

Minimising the same with reference to w_j yields

$$w_j^\star = \frac{\boldsymbol{y}\boldsymbol{z}_j^T}{(||\boldsymbol{z}_j||_2)^2} \ . \tag{4.14}$$

Plugging this into equation 4.13, the residual sum of squares is computed for each variable, and that variable incorporated into the selected subset which yields the minimum error among all the input variables.

This is the strategy used to select the first variable $\boldsymbol{z}_{(1)}$. If we now replace $y(k)$ by $\{y(k) - w_{(1)}z_{(1)}\}$, and repeat the earlier process after eliminating $z_{(1)}$ from the set of input variables, we would be regressing in a subspace orthogonal to $z_{(1)}$. This is therefore tantamount to a Gram-Schmidt orthogonalisation with greedy variable selection; the former can be generalised into a QR decomposition process, e.g. the Householder rotation. The process has a variant in the Efroymson's Algorithm [37].

Efroymson suggested an important variation on forward selection. Here, after each variable (other than the first) is added to the set of selected variables, a test is made to see if any of the previously selected variables can be deleted without appreciably increasing the residual sum of squares. Efroymson's algorithm incorporates criteria for the addition and deletion of variables as follows :

Addition

Let $\mathcal{E}_p$ denote the residual sum of squares with p variables. Computing

$$R_a = \frac{\mathcal{E}_p - \mathcal{E}_{p+1}}{\mathcal{E}_{p+1}/(K-p-2)} \tag{4.15}$$

if $R_a > \eta_a$, the $(p+1)^{th}$ variable is added to the set; for practical purposes, $\eta_a < 2$.

Deletion

Let $\mathcal{E}_p^s$ be the smallest residual sum of squares with p variables which can be obtained after deleting any variable from the previously selected set of variables. The ratio

$$R_d = \frac{\mathcal{E}_{p-1}^s - \mathcal{E}_p^s}{\mathcal{E}_p^s/(K-p-1)} \tag{4.16}$$

if $R_d < \eta_d$, the variable is deleted from the selected set. The procedure stops when no further additions or deletions are possible satisfying the above criteria.

It can be shown that a sufficient condition for the convergence of the algorithm is that

$$\eta_a > \eta_d \ . \tag{4.17}$$

However, there is still no guarantee that this algorithm will find the best-fitting subsets, although it often performs better than OLS (direct forward selection) when some predictors are highly correlated.

▮ *Basis Selection using OLS and Allied Methods*

Looking back at the original RBF formulation in equation 1.3, it can be observed that the output map $\boldsymbol{y}$ for an RBF is a multivariate linear regression on the derived variables z_i. Hence, variable selection for this linear regression is equivalent to basis function selection for the resultant RBF. The OLS technique has therefore been used in [40, 41] for FBF and RBF nets respectively for the step-by-step augmentation of the basis set. It is also evident that the entire bulk of statistical theory available on the topic, including Efroymson's Algorithm, can be adopted without modification for the purpose, and this constitutes a substantial advantage in favour of the regressional approach from the viewpoint of mathematical soundness.

4.2.2 Novelty-Based Selection of Basis Functions

The novelty-based approach to basis selection [39] uses clustering constructs. The main departure here from the rest of the models discussed in this chapter is that the set of basis functions is not available apriori, but is derived from the clustering process. It is the mechanism based on adaptive model size adjustment that calls for the incorporation of this model under the current section.

The basic method involves starting with an RBF with no basis functions, and allocating new nodes to it based on the *novelty* of the observations that arrive sequentially; where novelty requires the simultaneous satisfaction of two criteria

$$||\boldsymbol{x}_n - \boldsymbol{\mu}_{nr}||_2 > \epsilon_n \tag{4.18}$$

$$and \;\; e_n = ||y_n - f(\boldsymbol{x}_n)||_2 > e_{min} \tag{4.19}$$

where n denotes the iteration number, $\boldsymbol{x}$ the input, $\boldsymbol{\mu}$ the cluster centres, f the effective map and y the desired output (with subscripts interpreted accordingly). If an observation has no novelty, then the existing parameters of the system are adjusted using an LMS gradient-descent algorithm to fit that observation.

4.3 Basis Selection through Error-Complexity Trade-Off

In this section, basis functions will be selected based on the simultaneous penalisation of model error and complexity. Since the two need to be dealt with on a common basis, it is advantageous to either convert the model complexity into an error penalty or vice-versa. In the following sections, methods for basis function selection will be developed, first from the standpoint of the former, and then from that of the latter.

4.3.1 Converting Model Complexity into Error Penalty

One approach in linear regression is to use OLS-type algorithms in conjunction with a *model* order selector The model order selection is performed on-line using an

information criterion; the model order is steadily increased as long as the value of the information criterion goes on decreasing, and the process is terminated when this ceases to happen. In a sense, therefore, the importance of the information criterion is in terms of providing a terminating criterion to the OLS mechanism.

One such widely-used information criterion is the Akaike Information Criterion (AIC) [43] which can be stated as

$$AIC(k) = n \ln \hat{\sigma}_k^2 + 2k \tag{4.20}$$

where k is the model order, n the size of the training set and $\hat{\sigma}_k^2$ the least squares estimate of the residual variance σ_k^2. A detailed simulation of the AIC in linear regression can be found in [44]

4.3.2 Converting Error into a Complexity Penalty

In this subsection, a new entropy criterion (nonprobabilistic) is proposed. This is used to derive a suitable terminating condition for OLS, and for basis tree pruning in the case where the basis functions are hierarchically organised. The entropy criterion is similar to the one proposed in [46, 42] for error-free coding; a novel construct is used to incorporate the regression error variance into the above model.

In the following sections, starting with the original entropy model corresponding to the error-free case, the new entropy criterion will be derived, followed by extensions to the case of hierarchically ordered basis functions.

▮ *Entropic Measures of Model Complexity*

Let us suppose it is possible to express a function f as

$$f(x) = \sum_{i=1}^{N} w_i \psi_i(x) \ . \tag{4.21}$$

Further, we shall assume that $||\psi_i(x)||_2 = 1$. If $\psi_i(x),\ i = 1, \ldots, N$ are orthonormal, equation 4.21 may still be found to apply in the case we use projection pursuit to find successive w_is [45], i.e.

$$w_1 \ = \ < f(x), \psi_1(x) > \tag{4.22}$$

$$
\begin{aligned}
f_1(x) &= f(x) - w_1\psi_1(x) && (4.23)\\
w_2 &= < f_1(x), \psi_2(x) > && (4.24)\\
f_2(x) &= f_1(x) - w_2\psi_2(x) && (4.25)\\
w_3 &= < f_2(x), \psi_3(x) > \quad etc. && (4.26)
\end{aligned}
$$

As proposed by Coifman, Meyer and Wickerhauser [42, 46] for the orthonormal case, entropy

$$\mathcal{H}(f) = -\sum_{i=1}^{N} \alpha_i \ln(\alpha_i) \quad where \quad \alpha_i = \frac{w_i^2}{\sum_{i=1}^{N} w_i^2} \,. \tag{4.27}$$

Vetterli and Donoho [47] have remarked that this entropy criterion addresses coding complexity only, and not the associated error minimisation requirements. We shall proceed to make a construction that will remedy this

A New Entropy Measure

Let, at the k^{th} stage, the power of the approximated signal $\hat{f}(x)$ be given by

$$\hat{P} = \sum_{i=1}^{k} \alpha_i \tag{4.28}$$

Then the residual power is

$$\eta_k = 1 - \sum_{i=1}^{k} \alpha_i \tag{4.29}$$

assuming that the original signal has unit power. We shall now make the main construction by proposing that η_k is expressible as the sum of N_k basis functions of equal energy; it is worthwhile to remember in this regard that the given form of entropy function increases with the total number of terms, and is maximised under the assumption of equipartitions. With this, we have thus converted the error into a complexity penalty.

let us assume the following form for N_k :

$$N_k = C\eta_k^{\gamma} \tag{4.30}$$

where C and γ are constants. At the k^{th} stage, assuming that the analysed signal has unit power, the resultant entropy is given by

$$\mathcal{H}_k = -\sum_{i=1}^{k} \alpha_i \ln \alpha_i - \eta_k \ln \frac{\eta_k}{N_k} \,, \quad where \quad \eta_k = 1 - \sum_{i=1}^{k} \alpha_i \,. \tag{4.31}$$

Then, for the addition of the $(k+1)^{th}$ basis function, the increment in the entropy function becomes

$$\begin{aligned}
\Delta\mathcal{H}_{k+1} &= \mathcal{H}_{k+1} - \mathcal{H}_k && (4.32)\\
&= -\alpha_{k+1}\ln[C\alpha_{k+1}(\eta_k - \alpha_{k+1})^{\gamma-1}] \\
&\quad -\eta_k \ln\left[\frac{\eta_k^{\gamma-1}}{(\eta_k - \alpha_{k+1})^{\gamma-1}}\right] && (4.33)\\
&= -\eta_k\beta_{k+1}\ln\{C\alpha_{k+1}\eta_k^{\gamma-1}(1-\beta_{k+1})^{\gamma-1}\} \\
&\quad +\eta_k \ln\left\{\frac{1}{(1-\beta_{k+1})^{\gamma-1}}\right\} && (4.34)\\
\textit{where}\quad \beta_{k+1} &= \frac{\alpha_{k+1}}{\eta_k}
\end{aligned}$$

For $\gamma = 1$, the sizes of the partitions is fixed at $1/C$, while the number of partitions itself is allowed to vary. In this case,

$$\Delta\mathcal{H}_{k+1} = -\alpha_{k+1}\ln(C\alpha_{k+1}) \ . \tag{4.35}$$

In order for this quantity to be negative, the required acceptance condition for the $(k+1)^{th}$ basis function is that

$$\alpha_{k+1} \geq 1/C \ . \tag{4.36}$$

This specifies a threshold in terms of energy of the original signal. More importantly, the starting off from the terminating criterion in the approximating process (working backwards), we can substitute the value of C in the expression for entropy at the k^{th} step as

$$\mathcal{H}_k = -\sum_{i=1}^{k}\alpha_i \ln(C\alpha_i) + \ln(C) \ . \tag{4.37}$$

In the case where $\gamma = 0$, we have

$$\Delta\mathcal{H}_{k+1} = -\eta_k[\{\beta_{k+1}\ln(\beta_{k+1}) + (1-\beta_{k+1})\ln(1-\beta_{k+1})\} + \beta_{k+1}\ln(C)] \ . \tag{4.38}$$

This expression will be negative if

$$\beta_{k+1}\ln(C) > -\beta_{k+1}\ln(\beta_{k+1}) - (1-\beta_{k+1})\ln(1-\beta_{k+1}) \ . \tag{4.39}$$

If $\beta^{\star}_{k+1}$ is the solution to the above equation (replacing the '>' by an '='), the condition for entropy decrease is that

$$\beta_{k+1} > \beta^{\star}_{k+1} \quad \text{for } C > 1 \ . \tag{4.40}$$

In this case, the number of intervals having been fixed apriori, the evolved criterion dictates that the incremental energy accounted for by the new basis function must account for at least a minimum portion of the residual.

It has been found that the formulation evolves meaningful criteria only when $0 \leq \gamma \leq 1$. For internal points in this range, the resultant effect is a combination of those at the extremities.

∎ *Applications to Basis Selection*

The previous discussion assumed the circumstances of function approximation. However, by a simple reinterpretation of the variables, the derived entropy function can be directly applied to basis selection. This is obtained by assuming, in the linear regression setup, that

$$\alpha_i = \frac{\mathcal{E}_{i-1} - \mathcal{E}_i}{\mathcal{E}_0} \tag{4.41}$$

where $\mathcal{E}_p$ denotes the residual sum of squares with p variables. The condition in equation 4.36 then provides a terminating condition for basis function selection through OLS. This was experimented with the Deterding vowel data set (Appendix A). The variation of the number of selected basis functions using OLS with the threshold $(1/C)$ is shown in Figure 4.1, and the output performance (in terms of % recognition) in Figure 4.2. The figures illustrate a smooth error-complexity trade-off using the terminating criterion derived from the proposed entropy measure.

∎ *Hierarchies of Basis Functions*

In the previous sections, we have acquainted ourselves with the methodology of selection of basis functions by the elimination of redundant bases. However, we have postponed till now the question of the resolution at which the basis functions are obtained. This has a direct relation with the average spread of the basis

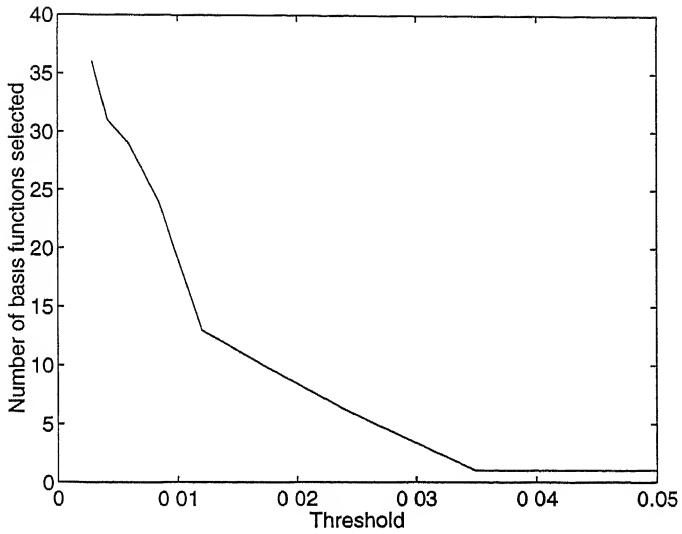

Figure 4.1. Variation of the number of basis functions evolved with the threshold on the minimal regressional contribution.

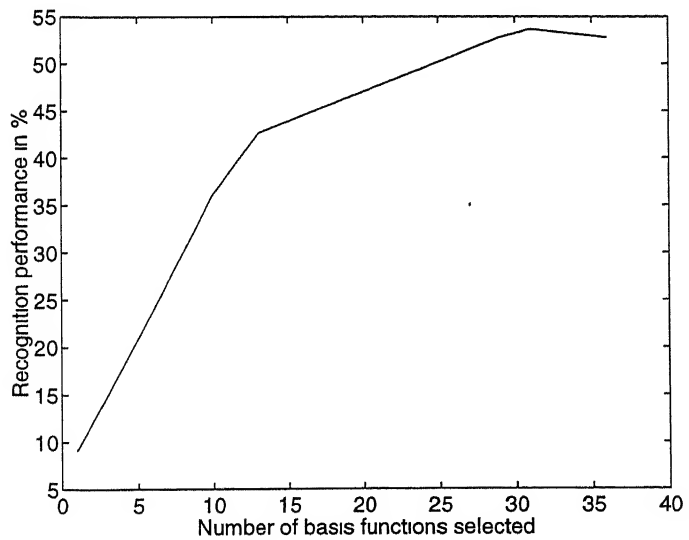

Figure 4.2: Plot of RBF model performance against the number of basis functions with OLS-based choice of basis functions.

functions specified; if the spread is low, the number of basis functions will be larger, and vice-versa. The above may also be looked at from the point of view of the error-complexity trade-off. If the spreads are lower, the approximation error will also decline, but the complexity of the model in terms of the number of basis functions used will increase. Had we used gradient descent for training, we could have estimated the means and the variances of the basis functions along with the weights, but the maladies of such a process have already been specified in chapter 1. Using the OLS method, on the other hand, requires all details of the basis functions to be fixed apriori so that the regression mechanism can operate independently of the previous step. However, the problem can be alleviated to some extent if the basis functions themselves occur at various levels of coarseness in a structured manner — in other words, if some of the basis functions at the finer levels are able to replace those at the coarser level.

Let us suppose the presence of a tree-structured set of basis functions where the basis functions associated with the internal nodes can be approximated by a linear combination of those associated with their children nodes. This is evidently the situation in the case of wavelet basis functions [48, 49] (these are extremely well suited for functional approximation, but have lower utility for PR operations because of their oscillatory nature) of several types; two important families are those of the Malvar wavelets and wavelet packets [42, 46]. Pictorially, the situation can be depicted as in Figure 4.3. Supposing the OLS operated on basis functions at the finest possible resolution, the question that we are going to pose at each node of the basis tree, working upwards from the leaf level, is as to whether we should need to replace the children nodes by the parent node. The decision in favour of the children nodes terminates the basis function replacement procedure for the concerned subtree. The strategy adopted for making the decision involves opining in favour of the option that entails a smaller complexity (including, if possible, an error penalty) so that the process of node replacement results in monotonic complexity reduction for the RBF. In the case where the basis functions at the finest resolution are orthonormal, the method of tree pruning through application of the entropy criterion is straightforward. However, if the above assumption is not valid (as,

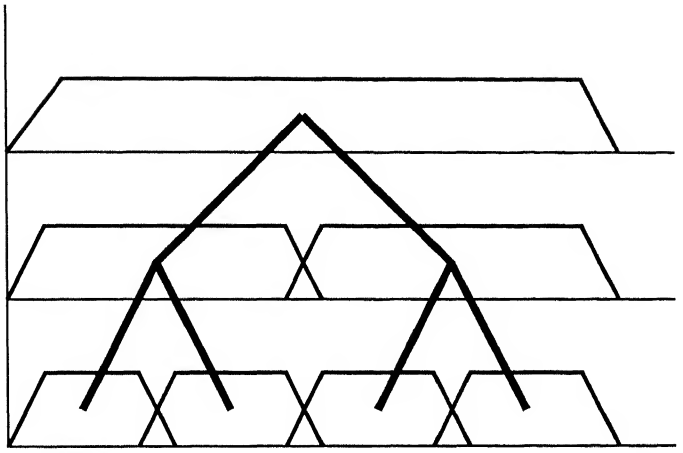

Figure 4.3: Pictorial depiction of a binary-tree structured hierarchy of basis functions; the abscissa roughly represents the support of the basis function, and the trapezia may be visualised as the envelope for wavelet basis functions.

e.g. for Gaussian basis functions), the method of basis function replacement has to operate concurrently with OLS. Every new basis function selected through OLS would entail an extra run up the basis tree. This is to ensure that the energy additivity property required by the entropy criterion and furnished by projection pursuit (or OLS) is not violated.

4.3.3 Applications in Data Compression

The tree-based basis selection procedure fits well into the framework of wavelets, and can be used in data compression along the lines obtained by Coifman, Wickerhauser and Meyer [46, 42]. The proposed entropy criterion may be seen as improvement to the entropy criterion developed by the above, since it is also able to work in case of lossy compression. The data compression angle, however, will not be pursued further in this thesis to avoid digression from the main topic.

Chapter 5

Modularisation of RBF Nets using Basis Neighbourhoods

In this chapter as well as in the next, techniques for modularising RBF networks will be developed. A modular neural network is formally defined as follows[1] :

> A neural network is said to be modular if the computation performed by the network can be decomposed into two or more modules (subsystems) that operate on distinct inputs without communicating with each other. The outputs of the modules are mediated by an integrating unit that is not permitted to feed information back to the modules. In particular, the integrating unit both decides how the outputs of the modules should be combined to form the final output of the system and which modules should learn which training patterns.

Modular networks usually combine supervised and unsupervised learning. Unsupervised learning takes the shape of competition among modules to produce the desired response. Once the module most suited for the purpose is identified, supervised training operates to modify the module parameters for superior performance. Modular networks have the advantages in that the training process becomes more efficient, since each module only handles a subpart of the total problem, and therefore constitutes a smaller network; and that internal data structuring becomes

[1] quoting Simon Haykin [7]

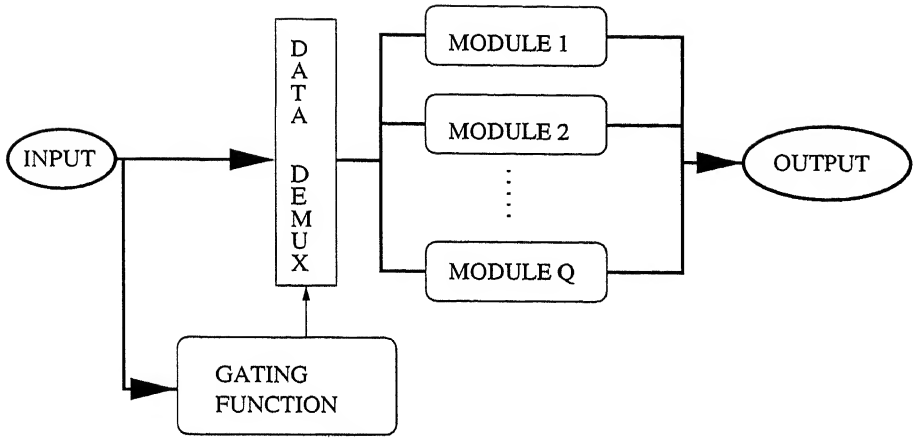

Figure 5.1: Schemata for the RBF-based modular ANN.

more pronounced so that the domains of operation of the different modules can be clearly identified. As an off-hand remark, structural modularity has a striking parallel in tissues in the biological context; these comprise cells of similar origin that are *differentiated* for different functions.

In the current discussion, it will be assumed that the integrating unit chooses one amongst the different modules to produce an output, an operation that will otherwise be referred to as *gating*. The gating unit demultiplexes the input data on the basis of a gating function, which identifies the input domain and accordingly allots an appropriate module for processing. The resulting modular configuration, which is a slight variation of the modular networks defined by Haykin [7] is illustrated in Figure 5.1.

This chapter will explore methods for the modularisation of RBF nets by concentrating on the basis functions. Since the basis functions are frequently obtained by clustering, a structure will be imposed on the evolving clusters, so that the final basis functions can be modularised using the same structure. Two graph structures that are adopted in this regard are the balanced tree structure and the flat two-dimensional grid structure. In the following sections, it will be shown how such structures can be realised through LBG and SOM based clustering respectively, thus leading to the desired modular architectures. In either case, the crux of the method rests on the identification of a neighbourhood in the input measurement

space with each module.

5.1 Graph-Associated Basis Functions

A *graph* $\mathcal{G}$ consists of a set of vertices v_i, $i = 1, \ldots, n$, and a set of edges e_{ij} connecting nodes v_i and v_j; it is said to be *directed* if each edge has a definite orientation (i.e. from the origin node to the termination node).

A *tree* is a graph in which there is a unique path from every vertex to every other A *rooted tree* is a directed tree which has a unique root node P_0 from which each other node is accessible by one and only one path. The *leaves* of a directed tree are the vertices from which no edges originate. The vertex of origin of any edge in a directed tree is called a *parent*, and the terminating node the *child.* A *binary* tree is a directed tree where each node has two or no children. A binary tree which is symmetric is called a *balanced* binary tree.

A graph is said to be *planar* if it can be mapped onto a plane such that no two edges intersect at a point which is not a vertex of the graph. A two-dimensional grid is one such planar graph. Let the distance d^g_{ij} be defined on the graph $\mathcal{G}$ as the fewest number of edges to be traversed to reach v_i from v_j. Thus, $d^g_{ij} = 0$ if $i = j$, $d^g_{ij} = 1$ if an edge e_{ij} exists between them, and greater than 1 otherwise. Let the n-neighbours of a vertex v_i be defined as all the nodes v_j for which $0 \leq d^g_{ij} \leq n$. the 1-neighbours of a vertex v_i are therefore all those vertices v_j for which e_{ij} exist.

With this basic idea about graphs, an effort will be made to attach basis functions to the vertices of the graphs described above. This is easier to visualise if we recall that each basis function may be associated with a data cluster. Thus, the balanced rooted binary tree can be identified with a hierarchy of clusters, with each cluster at the root or intermediate nodes giving rise to two nested subclusters. Vertices on the planar grid, on the other hand, can be looked upon as a set of data clusters arranged in the order of spatial contiguity. To further consolidate these analogies as well as to derive tractable methods for the evolution of such structured clusters, a brief foray will be made into the field of *vector quantisation*, which, in crude terms, can be summed up as a data compression strategy having intimate links with data

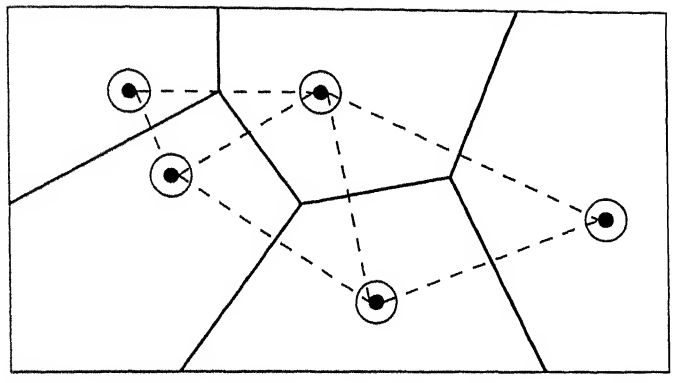

Figure 5.2: Vector quantisation of a 2-dimensional input space. The dark lines indicate partitions, and the circled dots the representative vectors.

clustering.

5.1.1 The Vector Quantisation Angle

The main purpose behind vector quantisation (VQ) [51, 52] is the approximation of the input space. In its simplest formulation, VQ aims to impose a partitioning of the input space based on minimum distance from a set of representative vectors. Under the assumption that there exists a distortion (distance) criterion $d(x_1, x_2)$ defined for any two vectors in the input space, the method aims to minimise

$$D = \frac{1}{2} \int_{R^n} d(\boldsymbol{x}, \boldsymbol{q}(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} \tag{5.1}$$

where $\boldsymbol{q}(\boldsymbol{x})$ is the value of the representative vector of the partition in which the vector $\boldsymbol{x}$ lies, and $f(\boldsymbol{x})$ is the multivariate probability density function of $\boldsymbol{x}$; minimising the criterion over each partition results in the evolution of the partition centroids as representative vectors as in Figure 5.2.

5.1.2 Evolving Basis Functions from Code Vectors

In a sense, the code vectors of a VQ operation represent cluster centres in the input space. Let each code vector $\boldsymbol{x}_i$ therefore be used to realise a basis function of the form

$$\tilde{z}_i(\boldsymbol{x}) = \exp[-d(\boldsymbol{x}, \boldsymbol{x}_i)] \, . \tag{5.2}$$

Further, if

$$d(\boldsymbol{x}, \boldsymbol{x}_i) = (||\boldsymbol{x} - \boldsymbol{x}_i||_2)^2$$

we may form a basis function

$$z_i(\boldsymbol{x}) = \exp[-\frac{(||\boldsymbol{x} - \boldsymbol{x}_i||_2)^2}{2\sigma^2}] \tag{5.3}$$

where σ^2 is the average variance of a cluster.

5.1.3 Tree Vector Quantisation (TVQ)

This technique for vector quantisation relies on the recursive splitting of clusters by the LBG algorithm [51, 53]. The method comprises first computing the centroid of the original cluster (the quantisation vector), and producing two child vectors by displacing the same by a small random vector in mutually opposite directions. Reclustering with Isodata (section 2.5.1) produces two stable clusters out of the original cluster, and the process may be repeated for a desired number of times to obtain a balanced binary tree of clusters.

- *Sensitivity Issues*

The main problem with the LBG algorithm is that since the initial perturbation at any cluster-splitting stage is random, it is possible for some of the splits to be grossly unbalanced, leading to large size variations in the evolved clusters. A strategy that might be adopted to alleviate this problem is to orient the perturbation along the direction of the principal eigenvector of the covariance matrix of the data comprising the parent cluster. The size of the tree has to be chosen in accordance with the desired number of leaf clusters; some amount of caution must be maintained to ensure that each of the evolved clusters has a minimum number of points.

5.1.4 Kohonen's SOM

For a vector quantiser as described in the earlier section, it is extremely difficult to aggregate nearest neighbour information in the sense of contiguous partitions for

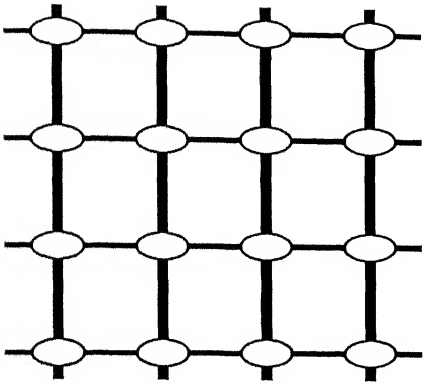

Figure 5.3: Square grid for node organisation.

a high dimensional input space. Kohonen's SOM [50, 54] is a vector-quantisation method in which the 1-neighbour computation requirement is eliminated by the apriori specification of the neighbourhood of a set of nodes which evolve into the required cluster centres. Let us suppose, for example, that the starting graph structure $\mathcal{G}$ is a square grid as in Figure 5.3. Initially, a set of random vectors are used to initialise the vectors attached to the nodes. During the training phase, for each training vector $\boldsymbol{x}$, the code vector having minimum distance from $\boldsymbol{x}$ is found out; next, all the neurons in its neighbourhood (e.g. the 1-neighbourhood) are updated as follows :

$$\boldsymbol{x}_i(n+1) = \boldsymbol{x}_i(n) + \eta(n)[\boldsymbol{x} - \boldsymbol{x}_i(n)] \; . \tag{5.4}$$

∎ *Issues Pertaining to SOM*

There are a great many ways SOMs can be modified in specific cases. These are :

1. The network topology : in this thesis, a 2-dimensional hexagonal topology has been favoured. This was because the square grid with four 1-neighbours was deemed to be sparse, while the same with eight 1-neighbours was found to be unstable.

2. The definition of a neighbourhood : here only 1-neighbours have been considered. The main reason behind this choice was the small size of the net.

3. The way the parameter η varies over the neighbourhood and over time : in this case, the value of η for 1-neighbours is about half the value for the central neuron, and η itself decreases over time from a value close to 1 to a small value; the above specification is largely empirical, and one amongst several alternative regimes.

The subjectivity of these issues arises from the fact that there do not exist rigorous proofs for SOM convergence even for linearly connected SOMs [55, 56, 57], and the complexities shoot up with the increase in dimensionality. With a greater laxity in expression, however, three prominent interpretations emerge [7] :

1. Approximation of the input space in the sense of vector quantisation.

2. Neighbourhoods on the SOM roughly correspond to neighbourhoods in the input space.

3. A rough estimate of the probability density of the input vectors is realised through the SOM.

5.2 Detailed description of TVQ-Based RBF Modularisation

The main idea in TVQ-based modularisation of RBFs is to identify subtrees at a particular depth in the LBG-derived VQ tree with RBF modules. The root nodes of the subtrees perform the function of gating, so that a datum is absorbed (during training as well as during field operation) by the module whose root wins the VQ process; the basis functions in an RBF module correspond to the clusters identified with the leaf nodes of the subtree. The idea is illustrated in Figure 5.4, which depicts a VQ tree of depth 2. The rectangles represent the two gating nodes, and the dotted lines outline the two modules. It is to be noted that different modules do not share basis functions.

The model building phase has two training procedures, one of which is unsupervised, and the other supervised. The former takes the shape of forming a VQ tree

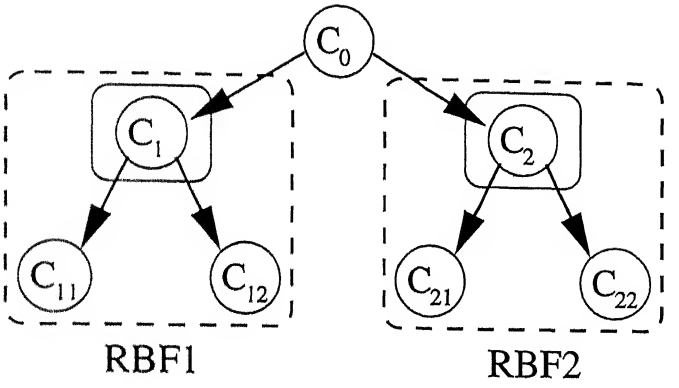

Figure 5.4: Deriving modules from a VQ tree

through recursive application of the LBG algorithm. Once the VQ tree is formed, the depth (from the root) of the gate nodes D_G is determined from the number of modules desired N_M by the simple relation

$$D_G = \log_2(N_M) \tag{5.5}$$

while the number of basis functions per module N_B is

$$N_B = 2^{(D_{VQT} - D_G)} \tag{5.6}$$

where D_{VQT} represents the depth of the VQ tree. N_M modules are finally formed, each equipped with a gating node corresponding to a tree node at depth D_G, and having N_B basis functions which are leaf nodes of the subtree with the gating node at the root. Now, each training data value is quantised at the level of the gating nodes, and the data is transferred to the corresponding module for supervised training of the weights of (the linear part of) the RBF.

During the field operation phase, the module selection part is identical to that for the test datum; only in this case, the selected module accepts the datum as input, and produces a corresponding output class value.

5.2.1 Process Description : Algorithmic Form

- *Training*

1. Perform the LBG clustering on the training data set to obtain a VQ tree.

2. Select gating nodes at a specified depth.

3. Form RBF modules, each comprising the corresponding leaf nodes as basis functions.

4. Distribute the training data values to the various modules depending on which gating node emerges the winner in the VQ operation.

5. With the allotted data, derive the linear map for each module by linear-inverse computation.

- *Testing*

1. For each input data value, allow the gating nodes to compete in the VQ operation.

2. Choose the module corresponding to the winning node.

3. Compute the output of the selected RBF module for the test input.

4. Declare the class corresponding to the largest output value.

5.3 Simulation

Simulations for the above procedure were performed using the Deterding Vowel Data Set (details in Appendix A). For the same, a VQ tree with 16 leaf nodes was developed. Two parallel implementations were executed — the first with a 16-basis-function RBF, and the second with two 8-basis-function RBF modules. The final recognition performances in the two cases was 44.16% and 40.26% respectively. It is observed that the modularisation leads to a small reduction in the performance. However, this difference is expected to disappear for larger problems where the number of leaf nodes is also larger, and a substantial number of them would merely contribute to the error variance.

5.4 Detailed Description of the SOM-Based RBF Modularisation

The model definition starts off by choosing the cardinality N of basis functions and the topology of the SOM. Using a training set, and the SOM training algorithm outlined in section 5.1.4, it is possible to find the code vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and the average standard deviation σ. So far, the mode of training has been unsupervised.

Now, for every node (with attached vector $\boldsymbol{x}_i$) let there be a matrix $\boldsymbol{L}_i$ of dimension $m \times (n_i + 1)$, where n_i is the number of neighbours on the SOM and m is the total number of output classes.

Supposing further that the outputs of the basis functions for an input are computed according to equation 5.3, it may be possible to form an $(n_i + 1) \times 1$ vector $\boldsymbol{z}_i$ comprising the outputs of the i^{th} node and its neighbours. This could then allow the computation of the output $\tilde{\boldsymbol{y}}_i$ as

$$\tilde{\boldsymbol{y}}_i = \boldsymbol{L}_i \boldsymbol{z}_i \ . \tag{5.7}$$

This will pave the way for the iterative estimation of $\boldsymbol{L}_i$ in the supervised training mode. Here again, some of the important features of modular neural networks [7] come into play. Appropriate credit assignment is achieved through intermodule competition on the basis of maximum activation; the training is performed for an input vector $\boldsymbol{x}$ only on the matrix associated with the winning basis function. The result is that out of the N possible output matrices, only one is allowed to update its parameters for any given training datum — the one associated with the winning basis function, and operating upon the outputs of the same and its neighbouring basis functions. Accordingly, the field operation is comprised of three successive processes:

1. Identification of the $\tilde{i}^{th}$ basis function with the highest activation.
2. Computation of the output vector using the winning basis function and its neighbours through the operation of the matrix $\boldsymbol{L}_{\tilde{i}}$.
3. Identification of the output class as the one corresponding to the maximal element of the output $\tilde{\boldsymbol{y}}$.

5.4.1 Process Description : Algorithmic Form

- *Training*

1. Assume N code vectors $\boldsymbol{x}_i$, $i = 1, \ldots, N$ associated to N nodes on a given grid (graph) structure.

2. For each node, enumerate the neighbouring nodes.

3. Set code vectors to random initial values.

4. Perform supervisory training on the SOM using equation 5.4.

5. For each node, estimate the vector $\boldsymbol{x}_i$ from the population of input data values allotted to it from the training set; estimate also the value of σ using the deviations from the respective means.

6. For each input data value,
   for each node i,
   compute the output of the basis function z_i.
   Find the node with the maximum z_i.

7. For every node i, use all the input-output data pairs allotted to it through step #6 to compute

$$\hat{\boldsymbol{L}}_i = \boldsymbol{Y}_i \boldsymbol{Z}_i^{+} \tag{5.8}$$

where $\boldsymbol{Y}_i$ is the matrix with the output vectors corresponding to allotted points as coloumns, and $\boldsymbol{Z}_i$ is the same constructed with the basis function outputs, and $\boldsymbol{Z}_i^{+}$ is the pseudo-inverse of $\boldsymbol{Z}_i$ computed as

$$\boldsymbol{Z}_i^{+} = \boldsymbol{Z}_i^T (\boldsymbol{Z}_i \boldsymbol{Z}_i^T)^{-1} \tag{5.9}$$

- *Testing*

1. For an input data value, compute all the basis function outputs z_i.

2. Identify the winning basis function as the one with maximal z_i.

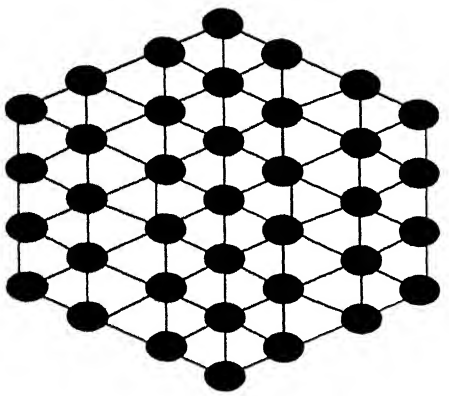

Figure 5.5: The nodes for an SOM on a hexagonal grid; each node has a maximum of six and a minimum of three neighbours.

3. Make the vector $\boldsymbol{z}_i$ using the outputs of the winning basis function and its neighbours.

4. Compute $\boldsymbol{y} = \hat{\boldsymbol{L}}_i \boldsymbol{z}_i$.

5. Declare the class corresponding to the largest element of $\boldsymbol{y}$.

5.5 Simulation

Simulations for the above procedure were performed using the Deterding Vowel Data Set (details in Appendix A). The input vectors were associated to 37 nodes on a hexagonal grid (Figure 5.5), and trained in the unsupervised regime over 200 iterations. In the next stage, supervised training was performed on the 37 associated matrices. It was observed that normalisation of vectors of basis functions defined over a neighbourhood with respect to the sum of the elements yields a superior performance for 2-neighbourhood-based basis selection.

The final recognition performance is listed in the table below :

Without Normalisation		*With Normalisation*		Direct Linear
1-neighbour	2-neighbour	1-neighbour	2-neighbour	
47.19%	38.96%	47.19%	45.67%	54.76%

Table 5.1: Performance of the SOM-based basis selection method. The rightmost coloumn indicates the situation where all the basis functions are included.

5.6 Comments

In this chapter, two interesting methods for modularising RBF nets were developed. In both cases, modules were roughly identified with neighbourhoods on the input space. For the first, the crucial idea was to use subclusters of the main cluster spanning a locality; in the second, a cluster and its graph neighbours were used to comprise the module, depending upon the Kohonen's SOM to ensure a rough correspondence of neighbourhoods on the graph with neighbourhoods on the input space. The SOM based modularisation method was found to be more robust of the two, possibly because of the inherent instabilities in the VQ tree derivation process. The modules in the former share basis functions, unlike the latter; this feature is expected to improve performance at the cost of increasing model complexity.

The gating functions are derived from the process of module synthesis, eliminating the requirement of artificial gating functions commonly found in modular neural networks [7] or complicated updating mechanisms associated therewith; on the other hand, the advantages of modular neural architecture, namely superior speed of learning, easy retraining and modular data organisation accrue naturally. Unsupervised and supervised training are also combined efficiently; the first is utilised to solve the *where* problem, essentially leading to rapid convergence into a specialised region on the neural graph, and the second to solve the *what* problem using the expertise available within the specialised domain. Thus, apart from providing an insight into one of the significant motives for basis selection, the proposed methods have wide-reaching implications, and certainly reveal themselves as worthy candidates for further investigation.

Chapter 6

Modularisation of RBF Nets using Piecewise-Linear Maps

In this chapter, linear unirank *atoms* of the form

$$\boldsymbol{W} = \boldsymbol{u}\boldsymbol{v}^T \tag{6.1}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are $m \times 1$ and $n \times 1$ vectors respectively, are used to structure the map from the intermediate vectors to the output class space in the RBF net, and derive a corresponding set of algorithms for training. In the process, the RBF net is modularised in much the same manner as illustrated in Figure 5.1. What is more interesting is the fact that the same graph structures — the balanced tree and the planar grid are once more used to structure the process of evolution of modules; in the former case, the modules evolve as the leaves of a multidimensional regression tree, while in the latter, these take the shape of a set of self-organising unirank matrices.

With reference to the form in equation 6.1, a couple of facts demand mention First, that any $m \times n$ matrix of rank r may be expressed as a sum of r such atoms using the singular value decomposition (SVD)

$$\begin{aligned} \boldsymbol{P} &= \sum_{i=1}^{r} \sigma_i \boldsymbol{w}_i \boldsymbol{v}_i^T \\ &= \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i \end{aligned} \tag{6.2}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{v}_i$ are the i^{th} unity norm left and right singular vectors and σ_i is the i^{th} singular value.

Next and more important, operation on a vector $\boldsymbol{z}$ with the matrix defined in equation 6.1 is equivalent to a succession of two steps :

- Computing the scalar

$$\xi = \boldsymbol{v}^T \boldsymbol{z} \tag{6.3}$$

  which produces the quantity ξ quantifying the correlation of $\boldsymbol{z}$ with $\boldsymbol{v}$. If it is assumed that the vectors $\boldsymbol{v}$ and $\boldsymbol{z}$ are normalised, ξ will necessarily lie in the range $[-1, 1]$. Hence, the value of ξ may be seen as reflecting the degree to which orientational localisation in the direction of $\boldsymbol{v}$ is satisfied by $\boldsymbol{z}$ (section 3.3.2).

- Attaching the scalar ξ to a vector $\boldsymbol{u}$ belonging to the output domain.

In other words, mapping with $\boldsymbol{W}$ defined in equation 6.1 is equivalent, in a less rigorous sense, to satisfying the *rule*

$$If \ < \text{aligned with } \boldsymbol{v} > \ then \ \ \boldsymbol{u} \ . \tag{6.4}$$

6.1 Structuring Linear Atoms

In the previous chapters, the linear map from the outputs of the (positionally localising) basis functions to the space of classes in R^m comprised an operation by a single matrix. Under the same assumption, the matrix $\boldsymbol{W}$ may be derived as per the least squares solution specified in equation 2.27. If a set of mutually orthogonal (i.e. the $\boldsymbol{v}_i$s and $\boldsymbol{u}_i$s are orthogonalised by lateral orthogonalisation mechanisms operating during the training phase as described in [9]) unirank atoms of the form in equation 6.1 are trained to approximate the desired input-output characteristics, their sum will converge to the least-squares solution mentioned previously in this section. Hence, the only gain in atomising the single matrix form may be in terms of tractability of the training problem for a large training set on a computational system with limited memory.

It is therefore clear that in order to gain functional advantage from structuring the above map, it would certainly be preferable to develop a *set* of linear maps to serve the same purpose; it is important to remember that the resultant map will be nonlinear. In essence, the methods will comprise evolving a series of compartments on the input space, each equipped with a unique linear map defined on it. The rank of the linear map will depend upon the procedure by which it is derived from training data.

In the following sections, two different approaches for developing modules on the input space (in this case, the intermediate space in an RBF i.e. the vector space formed by the span of the basis layer output vectors) are discussed in detail. The first involves recursive partitioning of the input space, so that the mapping mechanism takes the form of a regression tree. The other strategy stipulates a number of partitions (with flexible boundaries) and their neighbourhoods apriori, and proceeds to obtain a unit rank linear map for each partition.

For the regression tree, each module comprises a set of unirank atoms that lie along the path from the root of the tree to the leaf specific to the module, so that the aggregate map for the j^{th} module looks like

$$\boldsymbol{y} = \sum_{i=1}^{D} \boldsymbol{u}_{ji} \boldsymbol{v}_{ji}^T \boldsymbol{x} \tag{6.5}$$

where D is the depth of the regression tree. A number of modules thus share linear atoms. Gating is performed at each level of the tree by correlating with the corresponding $\boldsymbol{v}_{ji}$ vector used in implementing the piecewise nonlinearity.

In the other method of modularisation of the linear map, each module comprises a unit-rank matrix

$$\boldsymbol{W}_j = \boldsymbol{u}_j \boldsymbol{v}_j^T \ . \tag{6.6}$$

The data is gated to the module for which the correlation with $\boldsymbol{v}_j$ is maximum. In this case, there is no sharing of linear atoms.

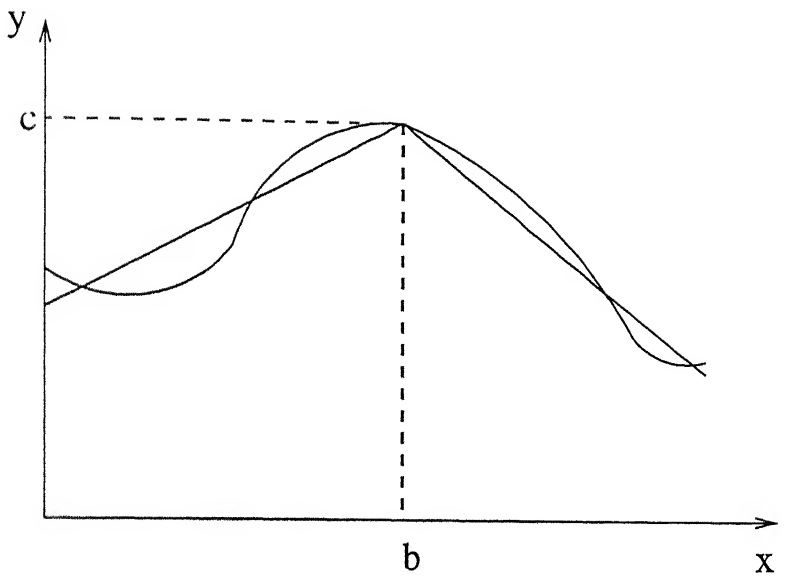

Figure 6.1: Piecewise fitting of a curve with a pair of straight lines.

6.2 The Regression Tree

The basic idea of the regression tree arises from the notion of piecewise linear maps. The idea was proposed for the unidimensional case in [58, 59]. The main construct is to approximate a data set specified in terms of input-output pairs (x_i, y_i), $i = 1, \ldots, K$ with a function constructed from a pair of straight lines as follows

$$\begin{aligned} y &= m_1(x - b) + c \quad if\ x < b \\ &= m_2(x - b) + c \quad otherwise. \end{aligned} \tag{6.7}$$

As can be easily observed from Figure 6.1, the function y remains continuous at $x = b$, although it ceases to be differentiable at this point. Extending this affine form into p dimensions, we have

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{u}_1(\boldsymbol{v}^T\boldsymbol{x} - b) + \boldsymbol{c} \quad if\ \ \boldsymbol{v}^T\boldsymbol{x} < b \\ &= \boldsymbol{u}_2(\boldsymbol{v}^T\boldsymbol{x} - b) + \boldsymbol{c} \quad otherwise. \end{aligned} \tag{6.8}$$

It is immediately evident that the hyperplane $\boldsymbol{v}^T\boldsymbol{x} - b = 0$ divides the input space, and possibly the training data set into two halves, each of which is approximated by a unique affine form as in equation 6.8; the resultant function is continuous on the hyperplane in question, and has the value $\boldsymbol{c}$ at all points on it.

Supposing that the constant term c is removed by assuming it to be identically equal to zero (this step is largely to ensure conformity with the earlier paradigms, and will actually deteriorate the performance of the overall model), and the vectors v and x augmented to produce

$$\tilde{v} = \begin{bmatrix} v \\ b \end{bmatrix} \quad and \quad \tilde{z} = \begin{bmatrix} z \\ -1 \end{bmatrix}, \tag{6.9}$$

then,

$$\begin{aligned} y &= u_1 \tilde{v}^T \tilde{z} \quad if \;\; \tilde{v}^T \tilde{z} < 0 \\ &= u_2 \tilde{v}^T \tilde{z} \quad otherwise. \end{aligned} \tag{6.10}$$

This mechanism can be used to split the input domain $\mathcal{Z}$ into two parts $\mathcal{Z}_1$ and $\mathcal{Z}_2$. Concentrating on the region $\mathcal{Z}_1$, if w_i denote the set of errors for corresponding inputs z_i, the set of pairs (w_i, z_i), $i = 1, \ldots, K$ can be used as per equation 6.10 to further split $\mathcal{Z}_1$ into $\mathcal{Z}_{11}$ and $\mathcal{Z}_{12}$ and so on. The approximator over $\mathcal{Z}_{12}$ will be given by

$$\begin{aligned} y_1 &= u_1 \tilde{v}^T \tilde{z} + u_{12} \tilde{v}_1^T \tilde{z} \\ &= (u_1 \tilde{v}^T + u_{12} \tilde{v}_1^T) \tilde{z} \end{aligned} \tag{6.11}$$

with u_{12} and $\tilde{v}_1^T$ arising from the latter partitioning mentioned above. It is easy to see that the above expression is in the same form as equation 6.2. Each operation on the data set by the atomic model of the form in equation 6.10 can be represented by a node with two branches as in Figure 6.2; recursive application of the same results in a binary tree. Each leaf of the final binary tree represents the smallest ultimate partition over the input space, which is accompanied by a resultant linear operator of the form in equation 6.2 assembled out of unirank linear atoms that lie en-route from the root of the tree to the leaf.

6.2.1 Training

The biggest hurdle in designing a regression tree lies in optimising the splits at each level. An incremental training scheme for the same can be derived through

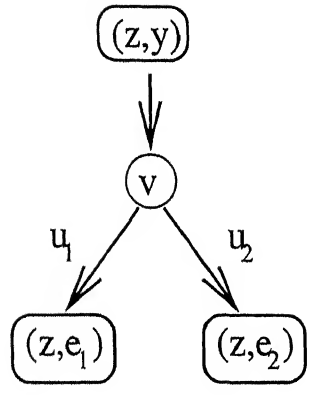

Figure 6.2: Data set split at a single node.

a gradient-descent approach, provided the function in equation 6.10 is rendered differentiable. This can be accomplished by rewriting equation 6.10 as follows :

$$\boldsymbol{y} = \boldsymbol{u}_1\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{z}}[1 - g_\infty(\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{z}})] + \boldsymbol{u}_2\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{z}}g_\infty(\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{z}}) \tag{6.12}$$

$$\textit{where} \quad g_\infty(a) = 0 \ \textit{if} \ a < 0 \tag{6.13}$$
$$= 1 \ \textit{otherwise.}$$

The function $g_\infty(\cdot)$ can be approximated by the softened function

$$g_\gamma(a) = \frac{1}{1 + e^{-\gamma a}}, \quad \textit{where} \ \ \gamma > 0 \, . \tag{6.14}$$

$g_\gamma(\cdot)$ represents a sigmoid as in Figure 2.1. Computing the differentials and finally taking the limit as $\gamma \to \infty$, we derive the the incremental learning scheme described in the following subsection.

∎ *Algorithmic Incremental Learning Scheme*

$$\begin{aligned}
\textit{Compute} \ \ q &= \tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{z}} \\
\textit{Compute} \ \ \boldsymbol{y} &= \boldsymbol{u}_1 q \ \ \textit{if} \ \ q < 0 \ \ \textit{and} \\
&= \boldsymbol{u}_2 q \ \ \textit{otherwise} \\
\textit{Compute} \ \ \boldsymbol{e} &= \boldsymbol{d} - \boldsymbol{y}; \\
\textit{If} \ \ q < 0,
\end{aligned}$$

$$\boldsymbol{u}_1 := \boldsymbol{u}_1 + \eta_u q \boldsymbol{e} \tag{6.15}$$

$$\tilde{\boldsymbol{v}} := \tilde{\boldsymbol{v}} + \eta_{\tilde{v}} \boldsymbol{e}^T \boldsymbol{u}_1 \tilde{\boldsymbol{z}} \tag{6.16}$$

else,

$$\boldsymbol{u}_2 := \boldsymbol{u}_2 + \eta_u q \boldsymbol{e} \tag{6.17}$$

$$\tilde{\boldsymbol{v}} := \tilde{\boldsymbol{v}} + \eta_{\tilde{v}} \boldsymbol{e}^T \boldsymbol{u}_2 \tilde{\boldsymbol{z}} \ . \tag{6.18}$$

An important observation that can be made from equations 6.15–6.18 is that the updating relation is the same as in the case of training a single-rank matrix for a given input-output map; it is almost as though $\tilde{\boldsymbol{v}}^T \tilde{\boldsymbol{z}}$ divides the data into two parts which are used in training the unirank mapping matrices $\boldsymbol{u}_1 \tilde{\boldsymbol{v}}^T$ and $\boldsymbol{u}_2 \tilde{\boldsymbol{v}}^T$ respectively.

6.2.2 Implementational Hazards

Unfortunately, the simplicity of the above training algorithm is partially illusory. For one, there seem to be a number of local minima, and the above method is likely to lead into one of these. This, therefore, dictates a judicious choice of the initial values of $\boldsymbol{u}$ and $\tilde{\boldsymbol{v}}$. One method for the same is to initialise $\tilde{\boldsymbol{v}}$ to reflect a hyperplane in the $\mathcal{Z}$ space (or partition thereof) passing through the centroid of all the vectors in the partition; this then may be used to divide the data-points into two parts $(\boldsymbol{X}_1, \boldsymbol{Y}_1)$ and $(\boldsymbol{X}_2, \boldsymbol{Y}_2)$ and compute $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ according to the equation

$$\boldsymbol{u}_i = \frac{\boldsymbol{Y}_i \boldsymbol{q}_i}{\boldsymbol{q}_i^T \boldsymbol{q}_i} \tag{6.19}$$

$$\textit{where} \quad \boldsymbol{q}_i = \boldsymbol{Z}_i^T \boldsymbol{v} \ . \tag{6.20}$$

However, the resulting initial conditions are still random for all practical purposes, and several runs may be necessary to find a satisfactory split; this forms a bottleneck in the training process and increases the computational requirements substantially.

Then again, the step-size needs to be carefully adjusted to prevent the iterates from becoming unbounded. This becomes particularly difficult for the various levels on the regression tree, since the input variances fluctuate widely.

In spite of these precautions, the iterative procedure is frequently found to fail with a real data set. The most common reason is that the hyperplane defined by

$\tilde{v}_T\tilde{x} = 0$ often moves right to the edge of the input data set so that all the z data values used for training come to lie on one side of the partition. The cause for this seems to be the removal of the constant term c as mentioned in section 6.2. To realise the impact of the same, one may consider trying to force c to zero in Figure 6.1; it is not very difficult to see that any such fit will be largely specious. A quick solution would be to retain the same, and augment equations 6.15 and 6.16 (and similarly the equations 6.17 and 6.18) in the training procedure by

$$c = c + \eta_c e \,. \tag{6.21}$$

6.2.3 Simulation Results

The above algorithms were tested on the Deterding Vowel Data Set (details in Appendix A). The output class information was expressed as an eleven dimensional vector (there being eleven classes). During the testing regime, the decision was made in favour of the class corresponding to the element with the largest output value.

It was found that the initial condition for the training regime largely determines the efficacy of the overall process. In this sense, the results presented below should be used only to illustrate the general trends of the resulting structure with respect to the control parameters, and are certainly amenable to further improvement using different sets of initial conditions. The minimum number of training points for a leaf node of the regression tree was chosen as the main control parameter. How the same governs the number of evolved leaf nodes can be observed in Figure 6 3.

The performance of the trained model was found to vary with the depth of the regression tree as shown in Figure 6.4. The figure shows a curve which peaks at forty-two nodes (corresponding roughly to a tree depth of 5.39) and tails off thereafter. It is also to be observed that the degradation in performance with only twenty-two (corresponding to a rough tree depth of 4.46) leaf nodes is not substantial either. This can be interpreted in terms of increasing generalising capacity with additional branching pitted against the limited amount of training data. The network therefore fails to generalise with respect to the test data set beyond the peak point.

The peak performance is around 46.75%. This corresponds to a tree depth of 5.4, and should be treated as a rank 5 approximation. Incidentally, a direct

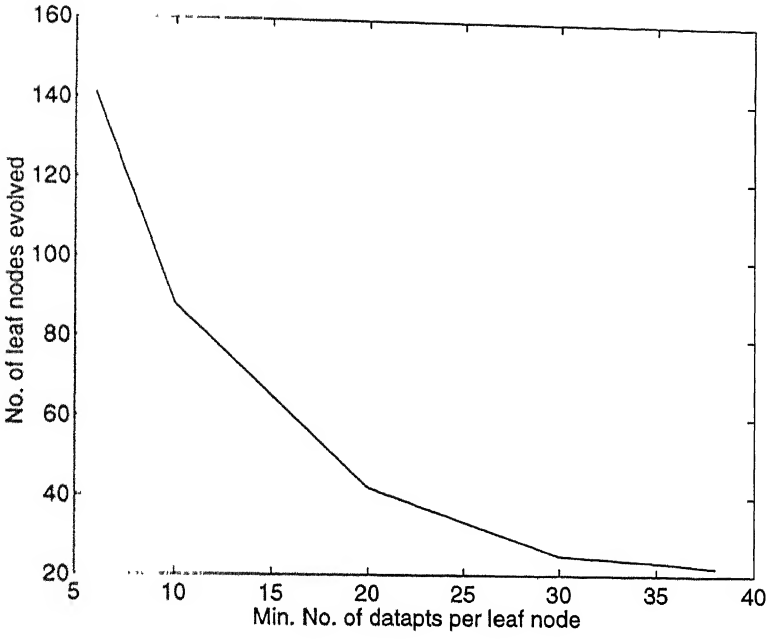

Figure 6.3: Variation in the number of evolved leaf nodes with the minimum number of training data values per leaf node.

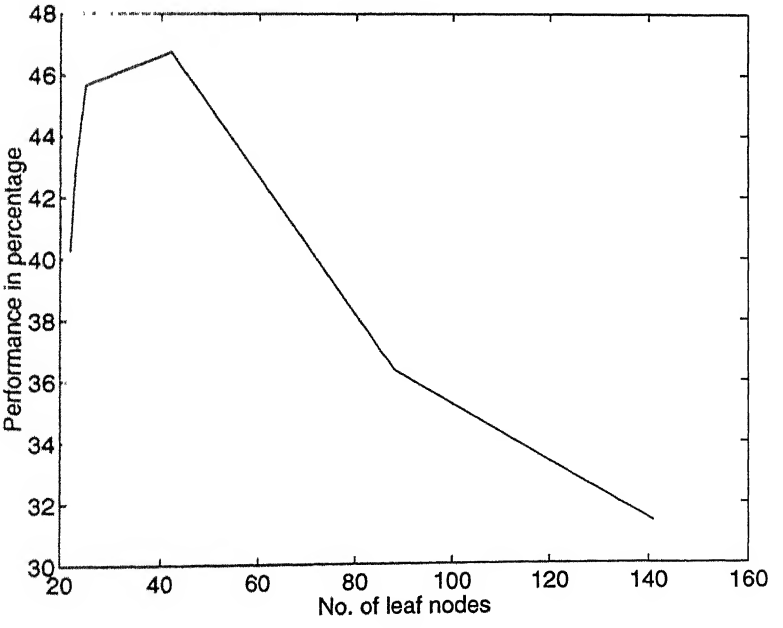

Figure 6.4: Regression tree performance trends for various number of leaf nodes.

linear map operating under identical preprocessing and trained optimally (as against the iterative solution mentioned above) using the Moore-Penrose pseudo-inverse provided a rank-5 performance of 42.42% and a rank-6 performance of 46.32%. With a larger training data set, the above method may be expected to outperform the same by a wider margin with sufficient optimisation of the training run.

6.3 Self-organised Unirank Linear Maps

In this section, some of the concepts that have been introduced in section 5.4 will once more be utilised, albeit in a slightly different perspective; as in the previous case, a planar grid will provide the basic skeleton for the proposed modular structure, and competition between nodes the basic method for module selection. However, where the inputs in the previous case belonged to the space of measurements on which positional basis functions were later developed, these here consist of the vector outputs of the basis layer as input to the subsequent linear map. Correspondingly, the vertices, instead of being attached to evolving cluster centres, are attached to unirank matrices; where the previous modes used distance-based PT matching for competition, the current nodes will use correlation-based OT matching for the same.

To expand on the above structure, let us start off with a graph $\mathcal{G}$ with vertices s_i and edges t_{ij} connecting vertices i and j. To each node s_i, are attached two vectors $\boldsymbol{v}_i$ and $\boldsymbol{u}_i$ such that the matrix $\boldsymbol{u}_i\boldsymbol{v}_i^T$ implements a map from the basis space $\mathcal{Z}$ to the output space $\mathcal{Y}$. The crucial idea is that the data $\boldsymbol{z}$ presented to the resultant structure is processed by the unirank linear map attached to the node s_k, for which the dot product $\boldsymbol{v}_k^T\boldsymbol{z}$ is maximum. This mechanism is used to distribute data for training to different modules, and invoke the appropriate module during field operation. Each module implements a unirank linear map during field operation; during the training phase, its parameters are updated using a gradient-descent algorithm as described in equations 6.24–6.25. The motivations for this mechanism are roughly the same as in the previous chapter; once again, the resulting modular map will allow interpretation and easy retraining.

6.3.1 Algorithms for Training and Field Operation

- *Training*

1. To nodes s_i on the hexagonal grid, attach random initial vectors $\boldsymbol{v}_i$ and $\boldsymbol{u}_i$.

2. Until convergence repeat

   - For every data value $(\boldsymbol{z}_k, \boldsymbol{y}_k)$, for every node s_i, compute

$$q_i = \boldsymbol{v}_i^T \boldsymbol{z}_k \ . \tag{6.22}$$

   - Find the winner node $s^\star$ with maximum q_i.
   - Update the winner node and its neighbours according to the equations

$$\begin{aligned} e_k^\star &:= \boldsymbol{y}_k - \boldsymbol{u}^\star \boldsymbol{v}^{\star T} \boldsymbol{z}_k & (6.23)\\ \boldsymbol{u}^\star &:= \boldsymbol{u}^\star + \eta_u q^\star e_k^\star & (6.24)\\ \boldsymbol{v}^\star &:= \boldsymbol{v}^\star + \eta_v e_k^{\star T} \boldsymbol{u}^\star \boldsymbol{z}_k & (6.25) \end{aligned}$$

     where η_u and η_v are functions of the path distance (along the graph) from the winner $s^\star$ and the iteration number, the considerations being the same as in section 5.1.4.

- *Field Operation*

1. Take a test datum $\boldsymbol{z}_k$ and compute q_i for all s_i.

2. Find the winning node $s_i^\star$ corresponding to the maximum value of q_i over all i.

3. Compute the output

$$\boldsymbol{y} = \boldsymbol{u}_i^\star \boldsymbol{v}_i^{\star T} \boldsymbol{z}_k \ . \tag{6.26}$$

4. Declare the class corresponding to the maximum element in $\boldsymbol{y}$.

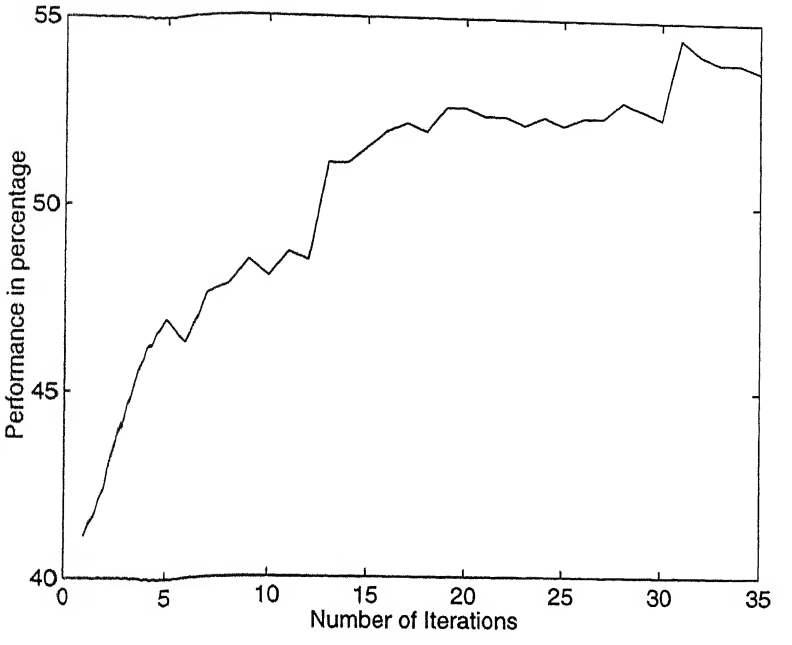

Figure 6.5: Iterative refinement of the linear self-organising map starting with random initial conditions The ordinate represents the performance of the trained model at an iteration on the test data.

6.3.2 Simulation Results

Simulations of the above model with the Deterding Vowel Data Set (details in Appendix A) yielded encouraging results. On account of the inherent random variability associated with the training process, different initial conditions result in different conical partitions of the input $\mathcal{Z}$ space. However, with η for the winning map reducing from the initial value of 0.1 exponentially with the iteration number and that of the neighbours decaying similarly from a lower value, the trained model provided sufficiently good results (i.e. over 50% correct decisions on the test set) from all runs. A typical training run is shown in Figure 6.5; it uses random initial conditions.

Using the output of one run as the input for another, it was immediately possible to derive yet higher performances for the test data set as illustrated in Figure 6.6. Overall, the highest performance obtained was a 56.49% recognition rate. It therefore becomes amply clear that the current atomic linear self-organising model is capable of delivering high performance on real data.

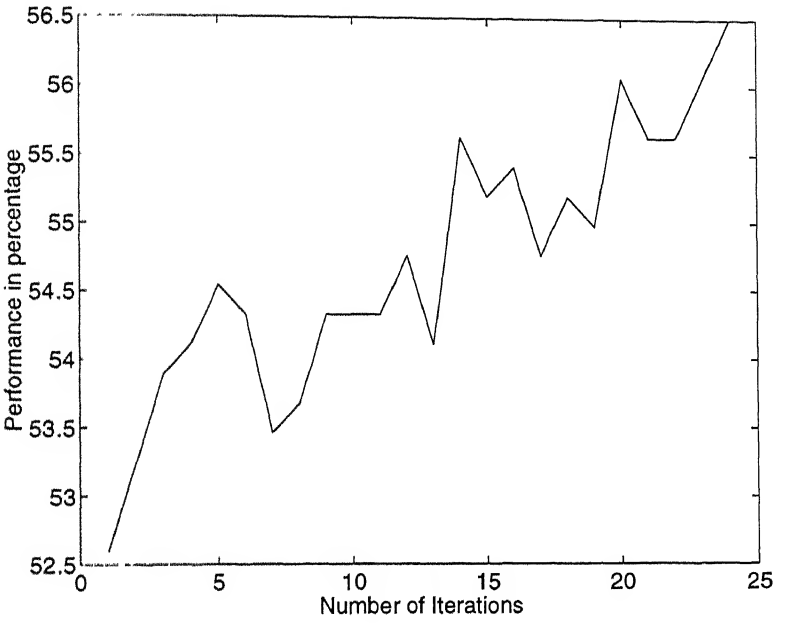

Figure 6.6: Iterative refinement of the linear self-organising map initialised with model parameters evolved from an earlier run. The ordinate represents the performance of the trained model at an iteration on the test data.

6.4 Comments

In this chapter two methods of modularisation of the RBF was obtained using an orientational-locality driven decomposition of the map from the intermediate vector (formed from the outputs of the basis functions) to the ouput class space into piecewise linear maps. The self-organising unit-rank structure was found to be more robust than the regression tree structure, particularly since in case of the former, the iterative updating of parameters is prone to instabilities as described in section 6.2.2. The former was also found to perform extremely well on the Deterding Vowel Data Set, and it seems likely that this is a result of the nonlinearity introduced into the modularisation process.

As a final comment on the topic of modularisation, it must be said that the interpretation of an RBF as a two-level matching process has been successfully utilised to modularise the RBF at each level; this reflects the extreme versatility of the model and highlights its potential in engineering design.

Chapter 7

Adapting the RBF for Speech Recognition

The field of speech recognition is concerned with the conversion of transduced and digitised articulated speech signals into a set of acoustic symbols. These symbols may be the simplest discernible speech units called phonemes, or ordered consonant-vowel pairs called diphones, or syllables, each of which is produced over a single breath wave, or even whole words or sentences. The former units are preferred to the latter because the number of tokens to be identified (and templates to be made thereof, assuming that some kind of a template-matching mechanism is adopted for PR) are fewer; this assuming that complete words can be identified through looking up a phonetic (or syllabic) dictionary. Since speech is the dominant mechanism for interpersonal communication between human beings, the importance of automatic speech recognition (ASR) in realising effective man-machine interfaces cannot be overemphasised.

Since the inception of the field of ASR almost half a century back, a number of groups around the world have worked on the problem, some of them backed by mighty corporations. The laboratory solutions that are available today are capable of transcribing clean speech from any speaker with average word error rates between 5% and 10% over large vocabularies of several thousand words [62], albeit at a very high computational cost. Performance degrades further if the ASR system is

presented with unfamiliar accents and words, high levels of background noise or acoustic reverberations. Currently marketed ASR systems are usually designed for high accuracies (above 98%) over restricted operating environments, e.g. for a small vocabulary, or for a single speaker.

It is however to be understood that the ASR problem has not been solved in any fundamental sense. The mechanisms for speech perception in humans are still not understood. The search for newer models has been on, driven partially by human curiosity and partially by the compulsion to improve the performance of existing ASR systems to a level comparable to that of the human auditory system.

In this chapter, a systematic study of the key issues in ASR will be made, and a solution proposed within the framework of the RBF. The first topic to be touched upon will be the time-frequency (TF) representation of articulated speech. Reasons will be cited in favour of adopting a spectrogram TF representation for the analysis of a speech segment, and a series of steps will be described to convert the same into a template comprising a sequence of acoustic feature vectors for use in an RBF. Special mention will be made of the role of the cepstral representation in accommodating problems relating to dynamic range minimisation and frequency translation of spectral vectors. Next, the prevalent ASR paradigms — which mainly pertain to acoustic vector sequence modelling and matching — like Hidden Markov Model (HMM), Dynamic Time Warping (DTW) and Time Dependent Neural Network (TDNN) will be described. Finally, a method for deriving time-warp-invariant templates will be proposed, starting with a note on the choice of the speech unit, and the same used in an RBF net to classify utterances corresponding to six different diphone classes.

7.1 Time-Frequency Representation of Speech

It is widely believed that the information in the speech signal is imbued in its time-varying spectral content. The starting point for most ASR systems is, therefore, the TF analysis [64] of speech signals. For this, a number of guidelines will be drawn from the explorations of Riley [63]. The most well-known TF distributions are :

- The *spectrogram* of a signal $x(t)$ given by

$$S_x(t,\omega) = \left| \int_{-\infty}^{\infty} g(\tau)x(t+\tau)e^{-j\omega\tau}d\tau \right|^2 . \tag{7.1}$$

  In other words, it is a two dimensional (2-D) depiction of the signal's short time spectra, with $g(t)$ as the analysis window.

- The *Wigner distribution* of $x(t)$ given by

$$W_x(t,\omega) = \int_{-\infty}^{\infty} x(t+\tau/2)x^*(t-\tau/2)e^{-j\omega\tau}d\tau . \tag{7.2}$$

The relationship between the two is that

$$S_x(t,\omega) = \frac{1}{2}W_g(t,\omega) * *W_x(t,\omega) \tag{7.3}$$

where $**$ denotes 2-D convolution [65].

Both these representations are shift-invariant, i.e. a shift in time or frequency of the signal should result in a corresponding shift in time or frequency in the transform. The spectrogram is a positive TF representation, while the Wigner distribution satisfies the marginals

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} W_x(t,\omega)d\omega = |x(t)|^2 \tag{7.4}$$

$$\int_{-\infty}^{\infty} W_x(t,\omega)dt = |X(\omega)|^2 \tag{7.5}$$

Riley shows that positivity and superposition properties of TF representations are strongly linked; where superposition is defined by the property that for any $\epsilon > 0$, there exists $\delta > 0$ such that

$$|F_{x+y}(t,\omega) - \{F_x(t,\omega) + F_y(t,\omega)\}| < \epsilon \ \ when \ \ |F_x(t,\omega)F_y(t,\omega)| < \delta . \tag{7.6}$$

The result is that if the TF distribution ceases to be positive, as for the Wigner distribution, cross terms will necessarily prove a problem for multi-component signals such as speech. It is for this reason that the spectrogram will be used for TF analysis in this thesis. There is, however, a flip side to the choice of spectrograms

in form of a bound on simultaneously attainable resolution in time and frequency. Supposing that for the window function $g(t)$ are defined

$$\sigma_T^2 = \frac{\int\int t^2 W_g(t,\omega) dt\, d\omega}{\int\int W_g(t,\omega) dt\, d\omega} \tag{7.7}$$

$$\sigma_\Omega^2 = \frac{\int\int \omega^2 W_g(t,\omega) dt\, d\omega}{\int\int W_g(t,\omega) dt\, d\omega} . \tag{7.8}$$

Then, it can be shown that

$$\sigma_T \sigma_\Omega \geq 1/2 . \tag{7.9}$$

Since the spectrogram can be obtained from the equation 7.3, the condition in equation 7.9 tantamounts to the bounds on the resolution mentioned above. An equality is obtained in equation 7.9 for the choice of a Gaussian kernel in equation 7.3. However, since a 2-D Gaussian is the Wigner distribution of a Gaussian pulse in the time domain, the above transform is equivalent to a spectrogram using a Gaussian window. Such a spectrogram will be the basic TF representation preferred in this thesis. The choice of a Gaussian window for the computation of the spectrogram has the additional operational advantage that it allows a high resolution DFT (say using 512 points) to be computed over a small effective window (of width twice the spread parameter σ).

7.1.1 The Sequence-of-Spectral-Vectors Representation

The TF analysis of a speech segment as per the guidelines stated above will yield a 2-D positive matrix, whose rows represent spectral bands, and the coloumns represent line segments; a typical speech waveform is shown in Figure 7.1, and its spectrogram in Figure 7.2. It is evident that the spectrogram matrix can be processed either as a 2-D image or as a 1-D sequence of spectral vectors. The author believes it is more appropriate to characterise the speech signal as a sequence of acoustic events, so the 1-D sequence representation mentioned above is used to build templates for incorporation in the RBF-based ASR implementation, with the interpretation that each acoustic event is characterised by a unique spectral vector.

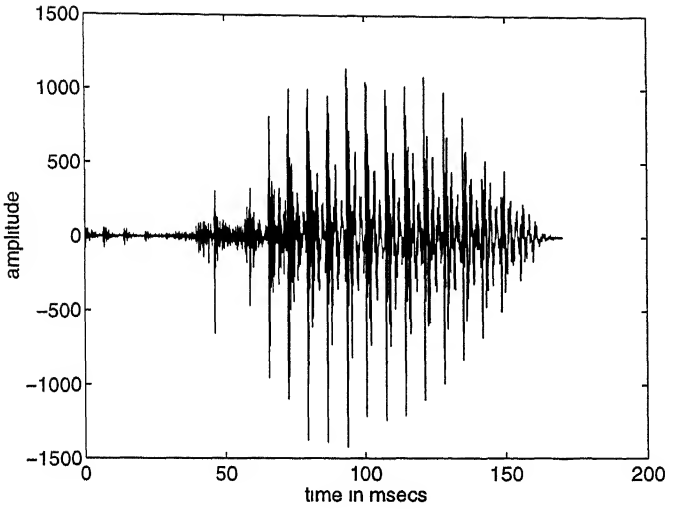

Figure 7.1: A segment of speech signal corresponding to the diphone 'gih'

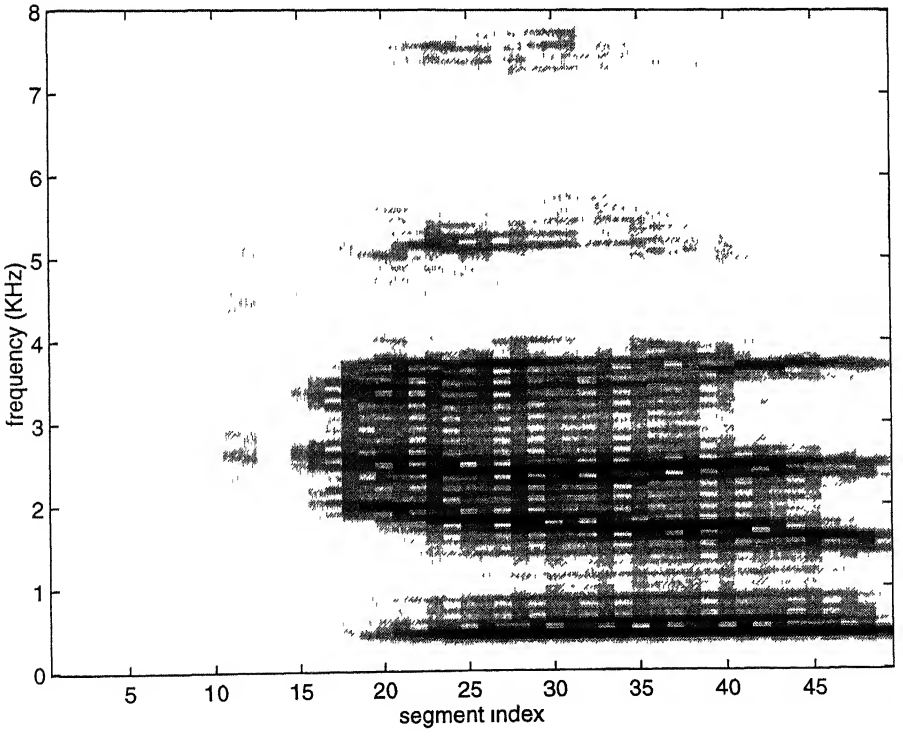

Figure 7.2: Spectrogram of the speech signal corresponding to the utterance 'gih' in Figure 7.1; it is to be noted that the values have not been log-transformed.

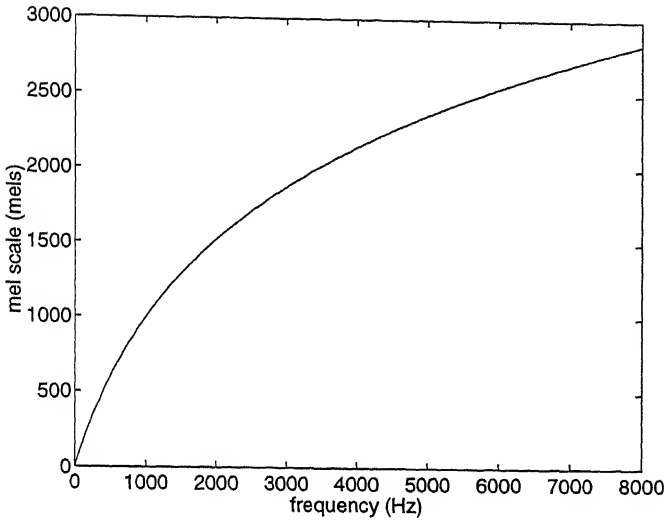

Figure 7 3. The Mel scale.

7.2 Processing the Spectral Vectors

Each spectral vector mentioned in the previous subsection has to be preprocessed before it can be used for pattern matching. Experiments in human perception have shown that the frequency resolution of the human ear decreases with increasing frequency. The spectral vector is therefore transformed onto a perceptual scale called the *Mel scale* (plotted against frequency in Figure 7.3) [66, 67] where

$$\text{Mel frequency} = 2595 \log_{10}(1 + f/700) \ . \tag{7.10}$$

This is performed by adding the energies over overlapping triangular windows placed linearly along the Mel scale. The resulting TF representation is as in Figure 7.4. A problem that is faced at this juncture is the large dynamic range of spectral vectors (three to four orders of magnitude), which jeopardises the use of weighted metrics as distance measures for matching, since the high-energy components corresponding to the formants of the voiced part (reflecting the resonant frequencies of the vocal tract) tend to nullify the contributions of the rest. A straightforward way in which this problem may be handled is by taking the logarithm of the elements of the TF matrix, which are positive by design; interestingly, the human perception of acoustic signals relates loudness to the log of the energy. However, elements having very small positive values produce large negative values, and have to be lower-thresholded. In

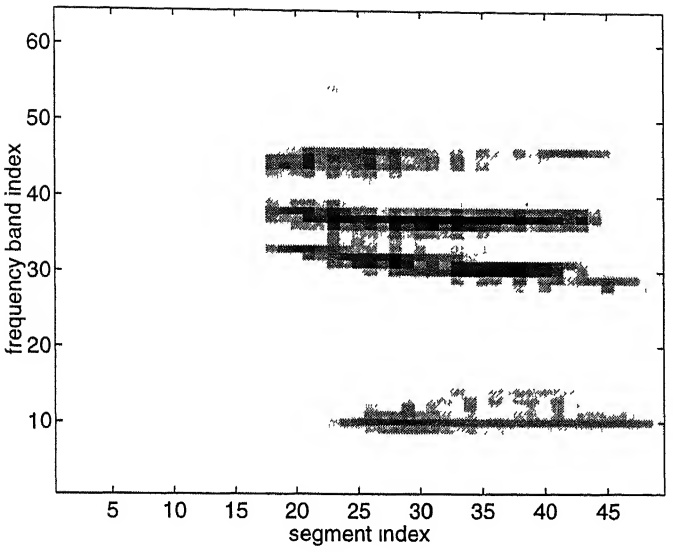

Figure 7.4· Mel scale frequency-warped spectrogram.

this thesis, the thresholding for a coloumn of the TF matrix is performed at 3 Bels (or 3 orders of magnitude) below the total energy of the coloumn. The TF representation thus produced is shown in Figure 7.5.

7.2.1 Cultivating Shift Invariance in the Spectral Vector

One of the major problems in using spectral vectors (coloumns of the TF matrix) to form templates for speech is that there are usually large variations in the spectra of utterances of the same speech unit by different individuals, and by the same individual at different times resulting from the variations in the vocal tract. A straightforward method of modelling this is as a linear translation along the perceptual frequency scale. It could greatly facilitate, therefore, to transform the spectral vector into a representation that is invariant to the linear shifts along the frequency scale. Let us consider a signal $u(t)$ with the Fourier Transform

$$U(\omega) = \int_{-\infty}^{\infty} u(t)e^{-\jmath\omega t}dt \ . \tag{7.11}$$

If $u(t)$ is translated by τ, the Fourier Transform of the translated signal becomes

$$\begin{aligned} U_\tau(\omega) &= \int_{-\infty}^{\infty} u(t-\tau)e^{-\jmath\omega t}dt \\ &= e^{-\jmath\omega\tau}U(\omega) \ . \end{aligned} \tag{7.12}$$

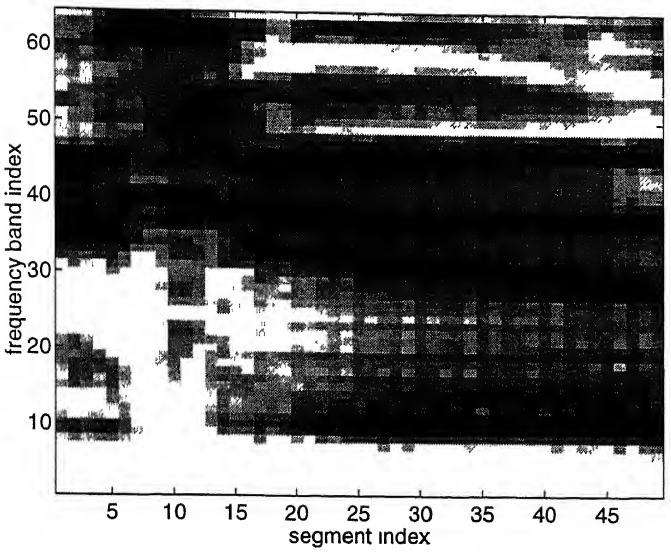

Figure 7.5: Frequency warped, log-transformed and thresholded spectrogram.

This means that the magnitude of the Fourier transform remains invariant to time translations. Thus, by taking the magnitude of the Fourier transform of the spectral vectors, a representation can be obtained which is invariant to translation along the frequency scale. The Fourier transform has the further consequence of decorrelating the spectral vector, so that an L_2 metric may be used for matching purposes.

It is interesting to note that the vector thus obtained is nothing but what is otherwise known as the cepstral representation of speech, which is portrayed in Figure 7.6. The same coefficients can be computed using an LPC or filter bank model [66, 61], and have been widely used in available ASR systems (although it is to be observed that the method of logarithmic thresholding may lead to significant differences). To add further discriminative power, the log-scaled energy values of the individual segments are also attached to the cepstral vectors to comprise what will be termed as acoustic feature vectors (AFV); the remaining problem pertains to making templates each of which has the form of a sequence of acoustic feature vectors, and devising methods for their matching.

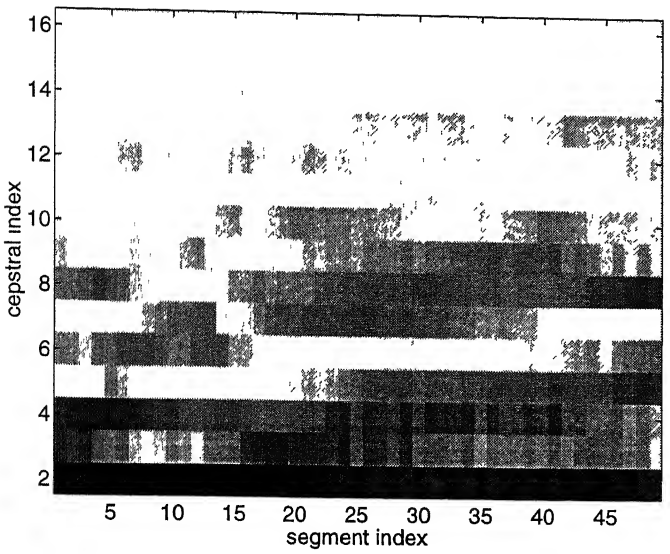

Figure 7.6: Intensity plot of the cepstral vectors

7.3 Acoustic Feature Vector Sequence Models and Matching Techniques

One of the major stumbling blocks in implementing ASR systems is the non-uniformly varying rate of speech articulation. The total time for the articulation of vowels (a few hundred milliseconds) is longer than, and has a greater variation than of stop consonants (a few tens of milliseconds). The dominant ASR models have implicit or explicit mechanisms for solving this problem as discussed in the following subsections.

7.3.1 Dynamic Time Warping (DTW)

ASR in this paradigm is by nearest neighbour classification, and template-matching is explicit. The central mechanism in DTW comprises matching a prototype sequence of acoustic feature vectors to a similar test sequence of possibly different length using a dynamic programming algorithm [68, 69, 70].

7.3.2 Hidden Markov Model (HMM)

An HMM is an autonomous stochastic automaton with a Markov chain state transition mechanism and outputs produced according to state-dependent probability densities. If $\boldsymbol{O}$ denotes an output sequence and $\boldsymbol{s}$ a state sequence, then the aposteriori probability

$$P(\boldsymbol{s}|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|\boldsymbol{s})P(\boldsymbol{s})}{P(\boldsymbol{O})} \; . \tag{7.13}$$

The word corresponding to the largest aposteriori probability of the state sequence is declared recognised. One important aspect to be noted here is that only the factor in the numerator of the r.h.s. of equation 7.13 is significant in relation to the classification operation. This again consists of two parts, the first of which, $P(\boldsymbol{O}|\boldsymbol{s})$ represents the contribution of the acoustic model (into which the templates are implicitly built), and the other $P(\boldsymbol{s})$, the contribution of the word or linguistic model; the former performs template matching by means of a stochastic dynamic programming algorithm called the Viterbi algorithm [71, 61, 72].

7.3.3 Time-Dependent Neural Network (TDNN)

This is an ANN architecture [9, 73, 67] with 3 or more processing layers. For the input, as well as for the intermediate stages, data is organised into a sequence of frames, with the dimensionality as well as the number of frames decreasing with each layer. The complete network is equivalent to a large MLP. The crucial architectural feature of the TDNN is that the hidden layers are expanded through the addition of multiple delay lines, one delay line for each of the original hidden units (this is what creates the sequence of frames). Since each of the layers can cope, within a limited scope, to time translation of the features handled by that layer, the aggregate network is able to handle moderate degrees of time-warping for the input speech pattern. The training is usually through the back-propagation algorithm used for MLPs, and the templates in the model are not explicit.

7.4 Forming Time-Warp Invariant Templates

The methods of ASR mentioned in the previous section perform time-synchronisation between the template and the test sample during the matching operation. In this section, a method is proposed by which time-warp-invariant templates are formed, so that the matching process is direct and computationally inexpensive. The technique rests upon the basic idea that the utterance corresponding to a speech unit comprises a sequence of acoustic events, each of which corresponds to a distinct acoustic feature vector, and is stationary over a minimum number of segments. The crux of the method lies in identifying the acoustic feature vectors (AFV) corresponding to the underlying acoustic events, and hierarchically organising them according to relative importance; this will be performed, in crude terms, by bunching up similar contiguous acoustic feature vectors to form temporally stable clusters.

7.4.1 Choice of the Speech Unit

In an ASR system, an important design criterion is the choice of the speech unit. The smaller the unit of speech, the fewer would be the number of templates to be made; this would, however, also imply smaller amount of contextual information embedded into each template, leading to error proliferation at the template-matching level and requiring powerful error correcting capabilities in the subsequent parser. A large unit of speech, on the other hand, will also imply a large number of templates, thereby adding to the training problem. Phonemes are the smallest acoustically distinguishable elements of speech, and have been used as the unit in several ASR systems. The problem with phonemes, however, is that the phenomenon of coarticulation (which, in crude terms, is a spill-over of neighbouring phonemes) severely mars their form, particularly for consonants, creating complications in implementing ASR systems. Choosing a larger unit like a diphone converts coarticulation into an advantage by using the same as contextual information; where each diphone consists of at most two contiguous phonemes, one of which is a vowel. A diphone is also the smallest unit of distinctly articulable speech. It is therefore the preferred unit of speech in this thesis.

7.4.2 Metric-Based Tree Representation of Waveforms

The proposed process of generation of the metric-based tree representation has two essential constituents :

- A hierarchical clustering scheme using a metric distance operating on the space of acoustic feature vectors. This is so that an acoustic event can be identified with a set of closely-valued AFVs.
- A sequential order constraint that prevents acoustic feature vectors having noncontiguous indices from forming clusters; this is to maintain the time sequence for AFVs.

The scheme operates upon a sequence of AFVs, each of which has a vector value and an index. Let us suppose that the original vector sequence is denoted by $\boldsymbol{x}_0[i], i = 1, \ldots, N$. The clustering algorithm can be specified as follows :

1. Assume each of the initial points to be a cluster.
2. At iteration n, where $n = 1, \ldots, N-1$, find two contiguous clusters $\boldsymbol{x}_{n-1}[j]$ and $\boldsymbol{x}_{n-1}[j+1]$ that are the closest, and merge them by replacing with their centroid.
3. If only one cluster is left, terminate; else go to the previous step.

The process thus reduces the number of AFVs in the sequence by 1 at each step, leading to a time-warped representation of the original sequence; the cluster merging sequence imposes a tree order on the clusters. As an example, the sequence shown in Figure 7.7(a) changes in subsequent steps into the sequences in Figures 7.7(b), (c) and (d) respectively. The corresponding tree structure is illustrated in Figure 7.8. The method bears close links with the technique of single-linkage clustering for a scalar ordered sequence [27], and topical similarities with the relational tree structure for waveform coding [75].

The tree structure certainly raises interesting prospects of incorporating syntactic constructs into the template-making and matching processes. It may be worthwhile,

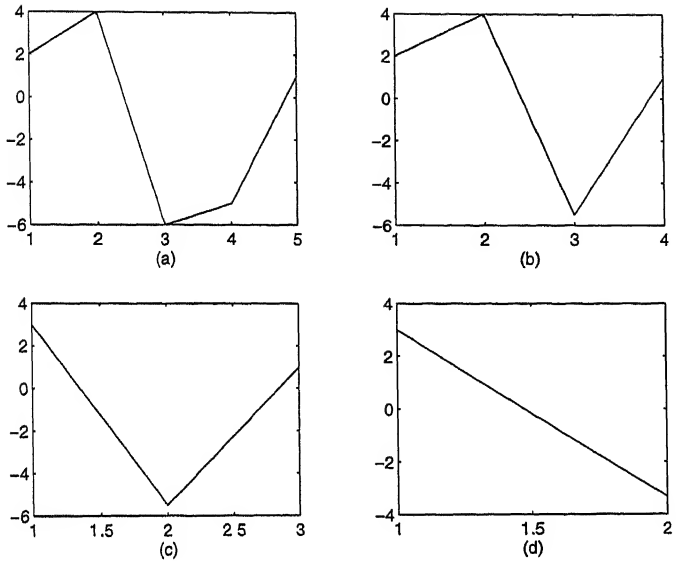

Figure 7 7: Clustered 1-D waveform at different stages.

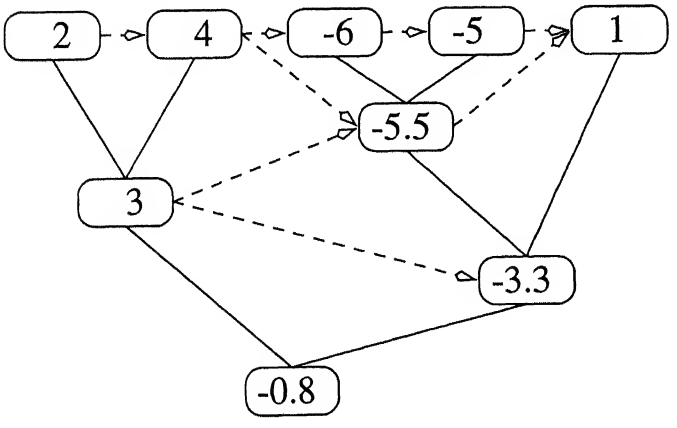

Figure 7.8: Tree structure corresponding to the clustering of the waveform in Figure 7.7. The numbers at the nodes denote the values of the sequence.

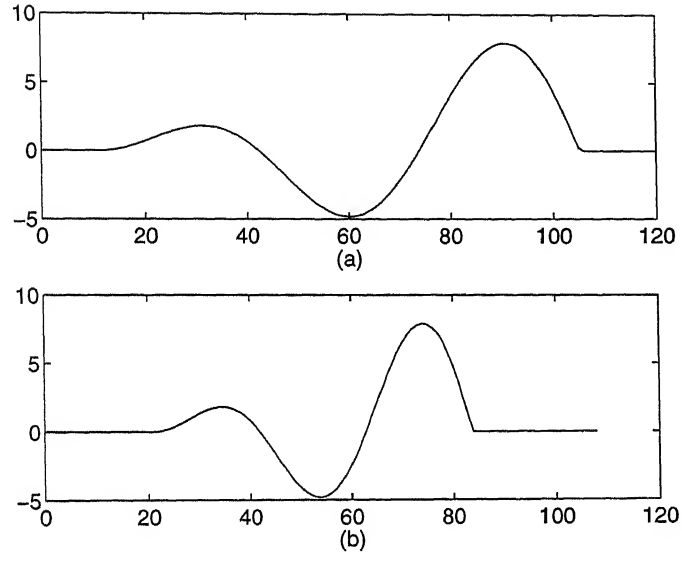

Figure 7.9: Variously shifted and dilated test signals of the form $\omega_i t \sin(\omega_i t)$, with $\omega_1 = 0.1$ and $\omega_2 = 0.15$.

for example, to consider elaborate tree clustering procedures to evolve sets of templates that have different internal tree structures. For the current quest, however, a direct, and somewhat crude technique is adopted to evolve templates from the linkage tree. This involves using the waveform obtained through aligning all the nodes at a certain depth of the linkage tree to represent the original waveform; in other words, truncating the original tree into a balanced tree of a certain specified depth. For the waveform specified in Figure 7.7(a), Figure 7.7(b) forms the depth-2 representation, while Figure 7.7(d) forms the depth-1 representation.

In spite of its apparent coarseness, the above representation does work well to provide shift and dilation invariance provided the truncation depth is chosen appropriately. As an example, the waveforms shown in Figure 7.9(a) and (b) (which are variously zero padded signals of the form $\omega_i t \sin(\omega_i t)$) have depth-2 distance-based tree encoding (DBTE) waveforms as in Figure 7.10(a) and (b) respectively.

The advantages of the proposed representation of waveforms are as follows :

1. All the representations are of equal length, equal to 2^d, d being the chosen depth of truncation of the distance-based tree. This will allow subsequent

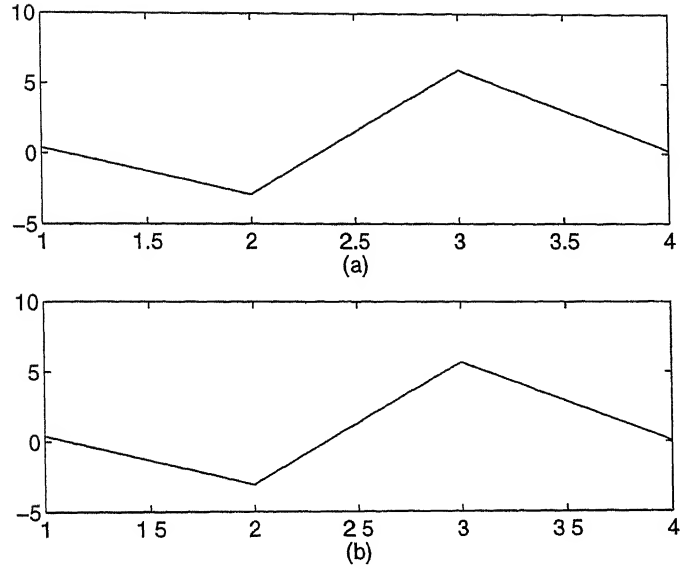

Figure 7.10: Respective depth-2 waveform representations for the signals in Figure 7 9.

clustering using the L_2 metric (or the Froebenius norm for vector sequences) using simple techniques like K-means or Isodata (described in section 2.5).

2. There is substantial data compression, leading to parsimonious template representation.

3. The L_2 metric (or correspondingly, the Froebenius norm) becomes a straightforward method of matching two sequence representations that can be directly used in the RBF formulation.

7.5 Implementation of the RBF-Based ASR Unit

An RBF-based ASR implementation has been made using diphones extracted from the TIMIT Database (details in Appendix B). Each diphone comprises one of the phones 'b','d','g','p', 't','k', followed by the vowel 'ih'. There are 9 male and 9 female utterances of each diphone in the training set and 5 male and 5 female utterances of each diphone in the test set. The utterances are randomised over different speakers.

First, for each digitised speech segment, sampled at 16kHz, the spectrogram is computed using 512 point FFT operating on a Gaussian window of spread $\sigma = 3$ms,

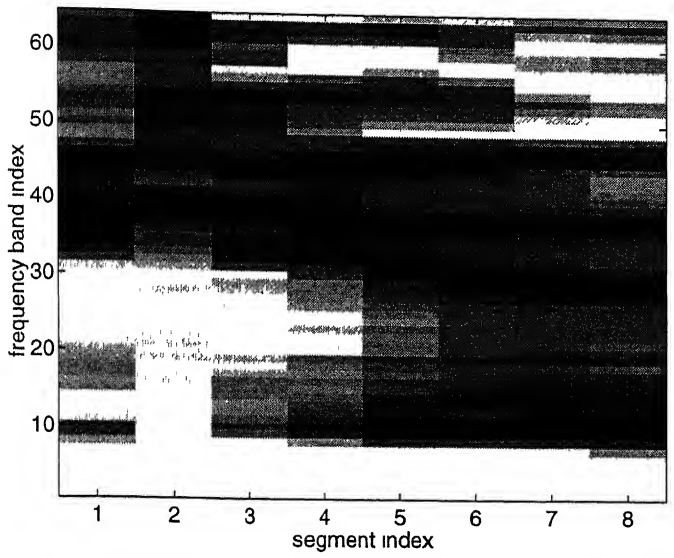

Figure 7.11: Depth-3 time-warp-invariant representations of the log scaled spectral vector sequence in Figure 7.5.

shifted by 3ms with each frame. Subsequently, the operations described in section 7.2 are performed to first obtain a sequence of 64-dimensional Mel-warped spectral vectors, and then the cepstral vectors from the same. The second to the sixteenth cepstral coefficient and the scaled (by 0.25) log energy value are then stacked to form an AFV.

The sequence of AFVs are next converted into a time-warp-invariant sequence of eight AFVs using the metric-based tree representation discussed in the previous subsection, choosing the truncation depth to be 3. These were subsequently used to form templates i.e. μ_is in the RBF, with the Froebenius norm used for distance computation. A one-shot unoptimised training using all the input data values as cluster centres (guided by sheer convenience, possibly hampering the network's generalisation capability) yielded a recognition performance of 60%. For comparison, templates were also made by the action of the time-warp-invariant transformation on the log-scaled spectrograms. In this case, under similar circumstances, the recognition performance was 56.67%. The action of a depth 3 truncated tree representation on the vector sequences in Figures 7.5 and 7.6 respectively are portrayed in Figures 7.11 and 7.12 respectively.

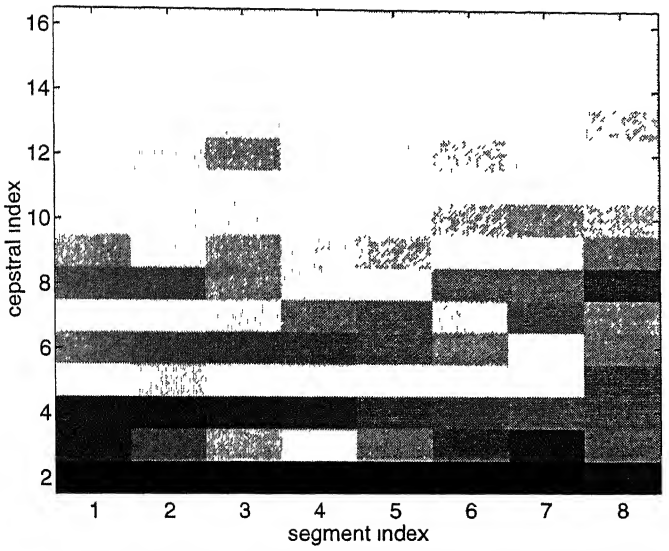

Figure 7.12: Depth-3 time-warp-invariant representations of the cepstral vector sequence in Figure 7.6.

7.5.1 Discussion

As is expected, the cepstral transform improves performance by inducing shift-invariance into the templates. The recognition performance of the ASR method outlined in this chapter, although not comparable with the state of the art, certainly merits further analysis. Apart from the rough-and-ready one-shot implementation enforced by the constraints of logistics, there are other factors which are possible culprits in lowering the performance. It has been found, for example, that the tree-based transformation may perform erratically if the depth is not properly chosen, or if some of the transitions in the AFV sequence are too abrupt. It is felt that further research on the metric-based tree representation of waveforms is required if its full potential is to be realised.

Chapter 8

Epilogue

In this final chapter of the thesis, some aspects of the work performed will be discussed. These include what might be deemed as personal contributions to the topic under discussion, the important conclusions drawn from the work, and the possibilities of extending the same for further research.

8.1 Personal Contributions

The following, in brief, are the personal contributions believed to have been made by the author in the course of the research work pertaining to this thesis, subject to his own knowledge of the available literature.

- Identification of RBF design criteria.
- Study of the effects of thresholding and normalisation of the outputs of basis functions on the performance of the RBF net.
- Interpretation of the RBF as a two-level template-matching process.
- Emphasising that the problem of basis function selection is equivalent to the problem of variable selection in linear regression referred to in statistical literature.
- Designing an information criterion which converts error into a complexity penalty for basis selection, and a study of the properties of the same.

- Modularisation of basis functions. This includes defining modularising structures and mechanisms and using LBG and Kohonen's SOM to synthesise modular RBFs.

- Modularisation of the linear map. This includes development of the self-organising piecewise linear functions of unit rank, and the extension of the unidimensional regression tree [58] to a multidimensional setup. The author has not been able to access [59], and as a result was not able to compare his results with later developments in the latter topic.

- Providing all the necessary tools and interpretations for building an exclusively RBF-based speech recognition system. A clarification of the utilities of the cepstral transformation was also made in the course of the stepwise design procedure.

- Designing a time-synchronisation technique to induce time translation and dilation invariance into speech templates to be used in the RBF-based PR system.

8.2 Conclusions Drawn from the Research Work

The most important conclusion that can be drawn from the performed work is that it is possible to control various aspects of RBF-based PR systems, enabling structured design of the same. It is expected that RBF systems will be used in adaptive signal processing with increasing regularity.

Another significant conclusion about the RBF is that its internal working can be interpreted, which paves the way for modularisation. Modularisation can be looked upon as the analogue of system decomposition for nonlinear trainable systems, and can prove valuable in system organisation.

In this thesis, it is assumed with respect to speech signals that a sufficient representation of the same can be made in terms of a sequence of *events*, each event being characterised as a segment with persistent traits. In this regard, a time-constrained distance-based tree encoding technique has been proposed, and the same

has been used to build a time invariant speech templates for use in an RBF. The performance of the resulting RBF for ASR using the above representation certainly does not conclusively prove the assumption of event-based characterisation of speech to be true, but provides evidence that more work organised along similar lines might produce a tractable solution to the problem of ASR of diphones or syllables.

8.3 Scope for Further Work

The RBF net provides a versatile tool for the design of adaptive systems, as described in section 1.5. In this thesis, aspects of the design of static RBF nets have been analysed without reference to sequence inputs. A possible extension of the work might therefore involve applying the same design guidelines to RBF-based nonlinear time-invariant adaptive filters. It may be recalled that the RBF can also be regarded as a linear map with nonlinearly transformed input variables; this interpretation clearly allows a foray into the nonlinear domain piggy-riding on linear domain tools and techniques. As an illustration, RBF based filters can be expected to be stable as long as the linear part of the same is stable, since the outputs of the nonlinear part are always bounded.

The problem of basis function selection has been dealt with in this thesis in the context of RBF design. However, the utility of the techniques derived in this regard extends over a wide range of problems, especially in data compression and signal representation, requiring further development of this angle. It would also be worthwhile to study how different families of (possibly nonlocal) basis functions may be suited for different applications in the context of signals and systems.

Finally, the idea of the distance-based tree structure used in speech representation needs to be further developed. The current implementation requires pruning of the same to a balanced tree of a certain depth, and in the process, incorporates only crude syntactic information; more sophisticated techniques were prevented by time constraints imposed by the curriculum. The performance of the evolved system certainly warrant further research on the design and analysis aspects of the proposed representation.

Appendix A

The Deterding Vowel Data Set

Summary : Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of LPC derived log area ratios.

Source :

- David Deterding (data and non-connectionist analysis)
- Mahesan Niranjan (first connectionist analysis)
- Tony Robinson (description, program, data, and results)

Maintainer : neural-bench@cs.cmu.edu

A.1 The Speech Data

(An ASCII approximation to) the International Phonetic Association (I.P.A.) symbol and the word in which the eleven vowel sounds were recorded is given in Table A.1. The word was uttered once by each of the fifteen speakers. The utterances if four male and four female speakers were used to train the networks, and those of the other four male and three female speakers were used for testing the performance.

vowel	word	vowel	word
i	heed	O	hod
I	hid	C:	hoard
E	head	U	hood
A	had	u:	who'd
a:	hard	3:	heard
Y	hud		

Table A.1: Words used in Recording the Vowels.

A.2 Front End Analysis

The speech signals were low pass filtered at 4.7kHz and then digitised to 12 bits with a 10kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters [60], giving a 10 dimensional input space.

Each speaker thus yielded six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the networks and 462 frames from the seven speakers used to test the networks.

A.3 Results

Table A.2 contains a summary of the results obtained by Tony Robinson and made available on the Internet.

Notes :

1. Each of these numbers is based on a single trial with random starting weights. More trials would of course be preferable, but the computational facilities available to Robinson were limited.

2. Graphs are given in Robinson's thesis showing test-set performance vs. epoch count for some of the training runs. In most cases, performance peaks at around 250 correct, after which performance decays to different degrees. The

Classifier	No. of hidden units	No. of correct units	Percent corrrect
Single-layer perceptron	-	154	33
Multi-layer perceptron	88	234	51
Multi-layer perceptron	22	206	45
Multi-layer perceptron	11	203	44
Modified Kanerva Model	528	231	50
Modified Kanerva Model	88	197	43
Radial Basis Function	528	247	53
Radial Basis Function	88	220	48
Gaussian node network	528	252	55
Gaussian node network	88	247	53
Gaussian node network	22	250	54
Gaussian node network	11	211	47
Square node network	88	253	55
Square node network	22	236	51
Square node network	11	217	50
Nearest neighbour	-	260	56

Table A.2: Results obtained by Tony Robinson on the Deterding data set.

numbers given above are final performance figures after about 3000 trials, not the peak performance obtained during the run.

Appendix B

The TIMIT Database

Contains : Segmented and labelled (at the phonetic, word and sentence levels respectively) speech data comprising articulated sentences chosen for the development and evaluation of ASR systems

Available as : The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) containing training and test data, distributed as the NIST Speech Disc CD1-1.1.

Source of the current information : File: readme.doc available on the above CD.

Brief Description : The TIMIT corpus of read speech has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. TIMIT has resulted from the joint efforts of several sites under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). Text corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI). The speech was recorded at TI, transcribed at MIT, and has been maintained, verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST). Additional information about the TIMIT Database, including the

referenced material and some relevant reprints of articles may be found in the printed documentation which is also available from NTIS (NTIS# PB91-100354).

Corpus Speaker Distribution : TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. A speaker's dialect region is the geographical area of the U.S. where they lived during their childhood years. The geographical areas correspond with recognized dialect regions in U.S. (Language Files, Ohio State University Linguistics Dept., 1982), with the exception of the Western region (dr7) in which dialect boundaries are not known with any confidence and dialect region 8 where the speakers moved around a lot during their childhood.

Corpus Text Material : The text material in the TIMIT prompts (found in the file "prompts.doc") consists of 2 dialect "shibboleth" sentences designed at SRI, 450 phonetically-compact sentences designed at MIT, and 1890 phonetically diverse sentences selected at TI. The dialect sentences (the SA sentences) were meant to expose the dialectal variants of the speakers and were read by all 630 speakers. The phonetically-compact sentences were designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest. Each speaker read 5 of these sentences (the SX sentences) and each text was spoken by 7 different speakers. The phonetically-diverse sentences (the SI sentences) were selected from existing text sources — the Brown Corpus (Kuchera and Francis, 1967) and the Playwrights Dialog (Hultzen, et al., 1964) - so as to add diversity in sentence types and phonetic contexts. The selection criteria maximized the variety of allophonic contexts found in the texts. Each speaker read 3 of these sentences, with each sentence being read only by a single speaker.

Suggested Training/Test Subdivision : The speech material has been subdivided into portions for training and testing. The criteria for the subdivision is described in the file "testset.doc" available on the CD.

References

[1] Fu K.S., *Introduction*, Digital Pattern Recognition ed. Fu K.S , Springer Verlag, 1980.

[2] Schalkoff R.J., *Pattern Recognition : Statistical, Structural and Neural Approaches*, John Wiley and Sons, 1992.

[3] Fukunaga K., *Statistical Pattern Recognition* 2^{nd} *ed.*, Academic Press, 1990.

[4] Fu K.S., *Syntactic Pattern Recognition*, Prentice Hall, 1982.

[5] Gonzalez R.C. and Thomason M.G., *Syntactic Pattern Recognition*, Addison-Wesley, 1978.

[6] Chen C.H. ed., *Proceedings of the IEEE Workshop on Expert Systems and Pattern Analysis*, World Scientific, 1987.

[7] Haykin S., *Neural Networks*, Macmillan, 1994.

[8] Pao Y.H., *Adaptive pattern recognition and neural networks*, Addison-Wesley, 1989.

[9] Kung S.Y., *Digital Neural Networks*, Prentice Hall, 1993.

[10] Kosko B., *Neural networks and fuzzy systems — a dynamical approach to machine intelligence*, Prentice hall, 1992.

[11] Mulgrew B., *Applying radial basis functions*, IEEE Signal Processing Magazine, Vol.13, No.2, March 1996, pp. 50–65.

[12] Cha I. and Kassam S.A., *channel equalization using adaptive complex radial basis function networks*, IEEE Journal on selected areas in communications, Vol.13, No 1, January 1995, pp. 122–131.

[13] Hush D.R. and Horn B.G., *Progress in supervised neural networks : what's new since Lippmann?*, IEEE Signal Processing magazine, January 1993, pp. 8–39.

[14] Baum E.B. and Haussler D., *What size net gives valid generalization?*, Neural Computation, Vol.1, 1989, pp 151–160.

[15] Cover T.M., *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, IEEE Trans. on Electronic Computers Vol.14, 1965, pp. 326–334.

[16] Park J. and Sandberg I.W., *Universal approximation usng radial basis function networks*, Neural Computation, Vol.3, 1991, pp. 246–257.

[17] Powell M.J.D., *Radial basis functions for multivariable interpolation : a review*, Technical Report DAMPT 1985/NA12, Dept. of applied mathematics and theoretical physics, Cambridge University, England, 1985. Also in Algorithms for Approximation, eds. Mason J.C. and Cox M.G., Oxford, 1987, pp. 143–167.

[18] Micchelli C.A., *Interpolation of scattered data : distance matrices and conditionally positive-definite functions*, Constructive Approximation, Vol.2, 1986, pp. 11–22.

[19] Poggio T. and Girosi F., *Networks for approximation and learning*, Proc. of the IEEE, Vol.78, No.9, September 1990, pp. 1481–1496.

[20] Specht D.F., *A general regression neural network*, IEEE Trans. on Neural Networks, Vol.2, No.6, 1991, pp. 563–576.

[21] Specht D.F., *Probabilistic neural networks*, Neural Networks, Vol.3, 1990, pp. 109–118.

[22] Parzan E., *On estimation of a probability density function and mode*, Annals of Mathematical Statistics, Vol.33, pp. 1065–1076.

[23] Kim H.M. and Mendel J.M., *Fuzzy basis functions : comparisons with other basis functions*, IEEE Trans on Fuzzy Systems, Vol.3, No.2, May 1995, pp. 158–167.

[24] Rumelhart D.E. and McClelland J.L. eds., *Parallel distributed processing : explorations in the microstructures of cognition, Vol.1*, MIT Press, 1986.

[25] Rosenblatt F., *The perceptron : a probabilistic model for information storage and organization in the brain*, Psychological Review, Vol.65, 1958, pp. 386–408.

[26] Minsky M.L. and Papert S.A., *Perceptrons*, MIT Press, 1969.

[27] Anderberg M.R., *Cluster analysis and applications*, Academic Press, 1973

[28] Everitt B.E., *Cluster Analysis (3^{rd} ed.)*, Edward Arnold, 1993.

[29] *Kohonen T., Learning vector quantization for pattern recognition*, Technical Report, TKK-F-A601, Helsinki University of Technology, Finland.,

[30] Albert A., Regression and the Moore-Penrose pseudoinverse, Academic Press, 1972.

[31] Haykin S., *Adaptive Filter Theory (2^{nd} ed.)*, Prentice hall, 1991.

[32] Geman S., Bienenstock E. and Doursat R., *Neural networks and the bias/variance dilemma*, Neural Computation, Vol.4, January 1992, pp. 1–58.

[33] Goffman C. and Pedrick G., *First course in functional analysis*, Prentice Hall of India, 1991.

[34] Hartman E.J., Keeler J.D. and Kowalski J.M., *Layered neural networks with Gaussian hidden units as universal approximators*, Neural Computation, Vol.2, No.2, 1990, pp. 210–215.

[35] Lindsay P.H. and Norman D.A., *Human Information Processing*, Academic Press, 1972.

[36] Edelman G.M., Neural Darwinism, Basic Books, 1987.

[37] Miller A.J., *Subset selection in regression*, Chapman and Hall, 1990.

[38] Hocking R.R., *The analysis and selection of variables in linear regression*, Biometrics, Vol.32, 1976, pp. 1–49.

[39] Yingwei L., Sundararajan N. and Saratchandran P., *A sequential scheme for function approximation*, Neural Computation, Vol.9, No.2, February 1997, pp. 461–478.

[40] Wang L.X. and Mendel J.M., *Fuzzy basis functions, universal approximation and orthogonal least-squares learning*, IEEE Trans. on Neural networks, Vol.3, No.5, September 1992, pp. 807–814.

[41] Chen S., Cowan C.F.N. and Grant P.M., *Orthogonal least squares learning algorithm for radial basis function networks*, IEEE Trans. on Neural Networks, Vol.2, March 1991, pp. 302–309.

[42] Meyer Y., *Wavelets*, SIAM, 1993.

[43] Akaike H., *Statistical predictor identification*, Ann. Inst. Math. Statist. Vol.22, 1970, pp. 203–217.

[44] Kundu D. and Murali G., *Model selection in linear regression*, Comput. Statist. & Data Analysis, Vol.22, 1996, pp. 461–469.

[45] Mallat S.G. and Zhang Z., *Matching pursuits with time-frequency dictionaries*, IEEE Trans. on Signal Processing, Vol.41, No.12, December 1993, pp. 3397–3415.

[46] Wickerhauser M.V., *Acoustic signal compression with wavelet packets*, Wavelets : A tutorial in theory and applications, ed. Chui C.K., Academic Press, 1992, pp. 679–700.

[47] Vetterli M. and Ramchandran K., *Best-wavelet packet bases in a rate-distortion sense*, IEEE Trans. on Image processing, Vol.2, April 1993, pp. 160–173.

[48] Zhang Q. and Benveniste A., *Wavelet networks*, IEEE Trans. on Neural networks, Vol.3, No.6, 1992, pp. 889–898.

[49] Bakshi B.R. and Stephanopoulos G., *Wave-net : a multiresolution, hierarchical neural network with localized learning*, AIChE Journal, Vol.39, No.1, 1993, pp. 57–81.

[50] Kohonen T., *Self-Organization and Associative Memory*, Springer Verlag, 1984.

[51] Gray R.M., *Vector Quantization*, IEEE ASSP Magazine, Vol.1, April 1984, pp. 4–29.

[52] Makhoul J., Roucos R., and Gish H., *Vector Quantization in Speech Coding*, Proc. of the IEEE, Vol.73, November 1985, pp. 1551–1558.

[53] Gray R.M., Buzo A., Linde Y., *An algorithm for vector quantization*, IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 84–95.

[54] Kohonen T., *The Self-Organizing Map*, Proc. of the IEEE, Vol.78, No.9, September 1990, pp. 1464 – 1480.

[55] Fort J.C., and Pages G., *About the Kohonen Algorithm : Strong or Weak Self-Organization?*, Neural Networks, Vol.9, No.5, 1996, pp. 773–785.

[56] Amari S.I., *Mathematical Foundations of Neurocomputing*, Proc. of the IEEE, Vol.78, No.9, September 1990, pp. 1443–1463.

[57] Amari S.I., *Field Theory of Self-organizing Neural Nets*, IEEE Trans. on Syst., Man and Cybernetics, Vol. SMC–13, Sept/Oct 1983, pp. 741–748.

[58] Friedman J.H., *A tree structured approach to nonparametric multiple regression*, Lecture notes in Mathematics, eds. Gasser Th. and Rosenblatt M., Springer Verlag, 1979, pp. 5–22.

[59] Breiman L., Friedman J.H., Ohlsen R. and Stone C., *Classification and Regression Trees*, Wadsworth and Brooks, 1984.

[60] Rabiner L.R. and Schafer R.W , *Digital Processing of Speech Signals*, Prentice Hall, 1978.

[61] Rabiner L.R. and Juang B-H., *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[62] Young S., *A review of large-vocabulary continuous speech recognition*, IEEE Signal Processing Magazine, Vol 13, No.5, September 1996, pp. 45–57.

[63] Riley M.D., *Speech time-frequency representations*, Kluwer Academic Press, 1989.

[64] Cohen L., *Time-Frequency Distributions · a review*, Proc. of the IEEE, Vol.77, July 1989, pp. 941–979.

[65] Claasen T. and Mecklenbrauker W., *The Wigner distribution : a tool for time-frequency signal analysis Part III*, Philips Journal of Research, Vol.35, 1980, pp. 372–389.

[66] Picone J.W., *Signal modeling techniques in speech recognition*, Proc. of the IEEE, Vol.79, No.4, April 1991, pp. 1215–1247.

[67] Nagakawa S., Shikano K., and Tohkura Y., *Speech, hearing and neural network models*, IOS Press, 1995.

[68] Sakoe H., and Chiba S., *Dynamic programming optimization for spoken word recognition*, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol.26, No.1, February 1978, pp. 43–49.

[69] Sakoe S., *Two level DP matching — a dynamic programming based pattern matching algorithm for connected word recognition*, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol.27, No.6, December 1979, pp. 588–595.

[70] Myers C.S. and Rabiner L.R., *A comparative study of several dynamic time-warping algorithms for connected-word recognition*, Bell System Technical Journal, Vol.60, No.7, September 1981, pp. 1389–1409.

[71] Rabiner L.R., *A tutorial on Hidden Markov Models and selected applications in speech recognition*, Proc. of the IEEE, Vol.77, No.2, 1989, pp. 257–285.

[72] Lee K.F., *Automatic speech recognition — the development of the SPHINX system*, Kluwer Academic Publishers, 1989.

[73] Waibel A., Hanazawa T., Hinton G., Shikano K and Lang K., *Phoneme recognition using time-delay neural networks*, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol.37, March 1989, pp. 328–339.

[74] Bengio Y. and De Mori R., *Connectionist models and their application to automatic speech recognition*, Artificial Neural Networks and Statistical Pattern Recognition, eds. Sethi I.K. and Jain A.K., North Holland, 1991.

[75] Shaw S.H. and Defigueiredo R.J.P., *Structural Processing of waveforms as trees*, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol.38, No.2, February 1990, pp. 328–338.